Self-Adjusting Networks

Stefan Schmid

“We cannot direct the wind,
but we can adjust the sails.”

(Folklore)

Acknowledgements:

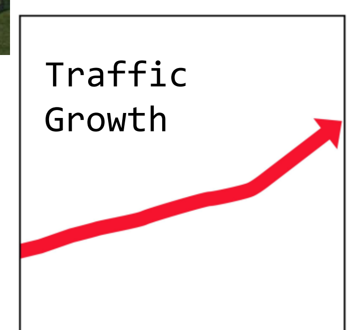
Trend

Data-Centric Applications

Datacenters ("hyper-scale")

+network

Interconnecting networks:
a **critical infrastructure**
of our digital society.



Source: Facebook

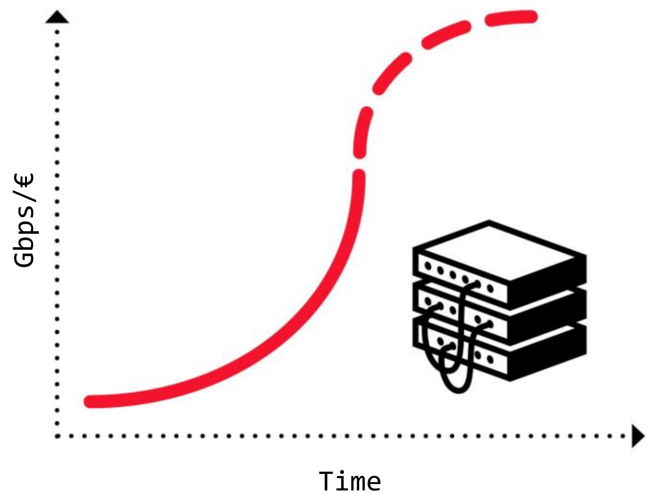
The Problem

Huge Infrastructure, Inefficient Use

⇢ Network equipment reaching capacity limits
  → Transistor density rates stalling
  → “End of **Moore‘s Law** in networking” [1]

⇢ Hence: more equipment, larger networks

⇢ Resource intensive and: **inefficient**



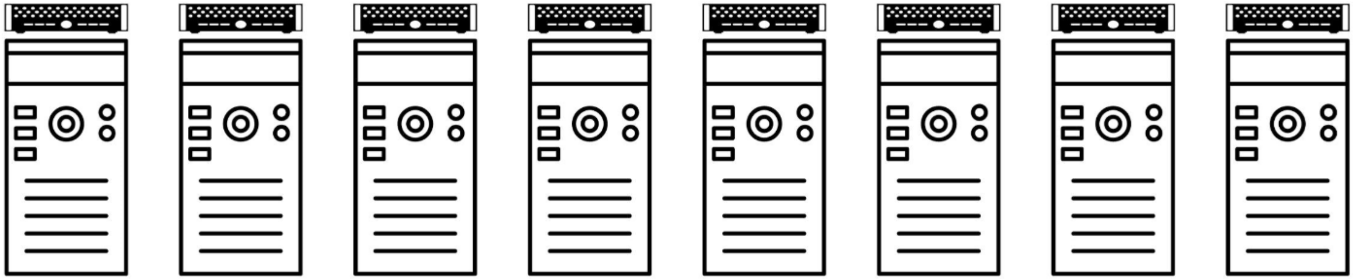
[1] Source: Microsoft, 2019

Annoying for companies, **opportunity** for researchers

Root Cause

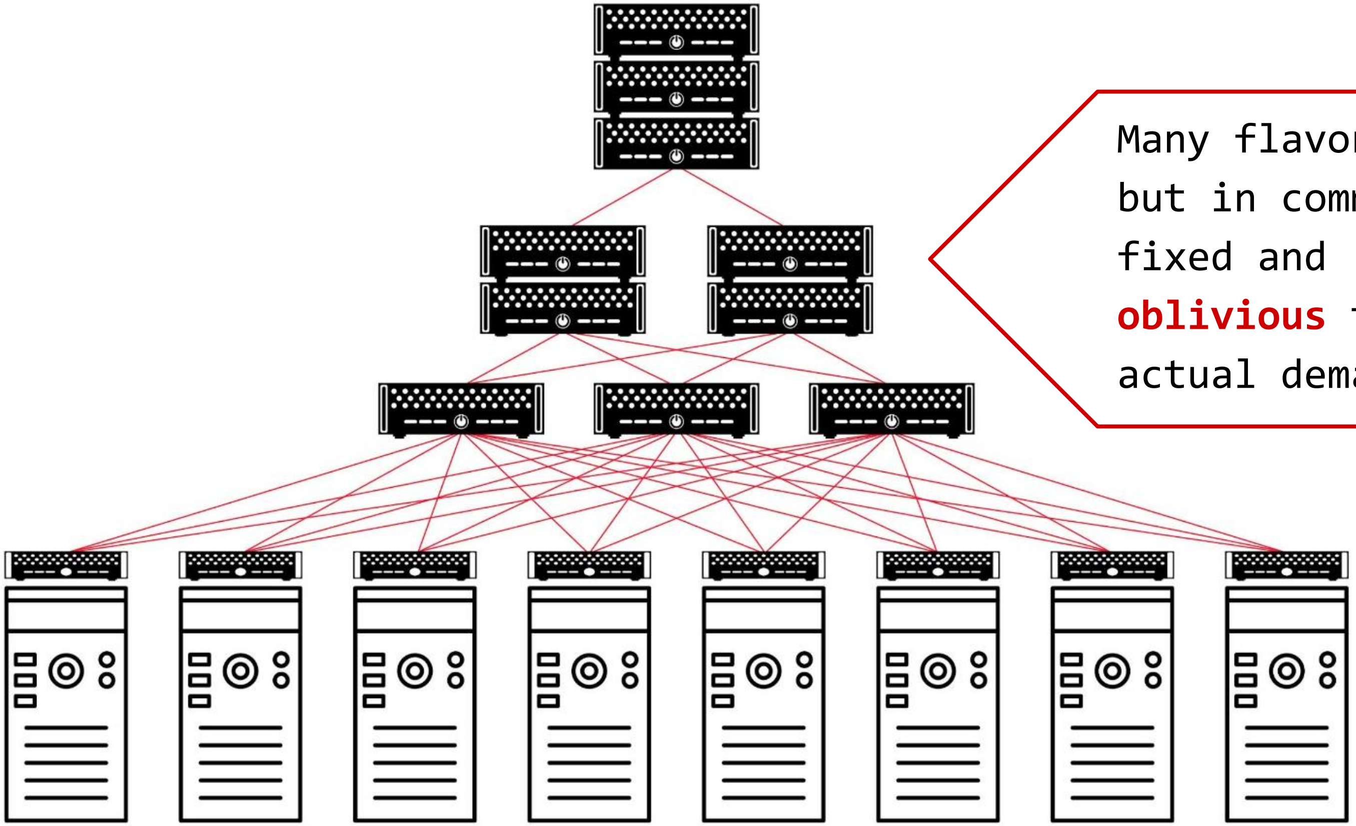
Fixed and Demand-Oblivious Topology

How to interconnect?



Root Cause

Fixed and Demand-Oblivious Topology

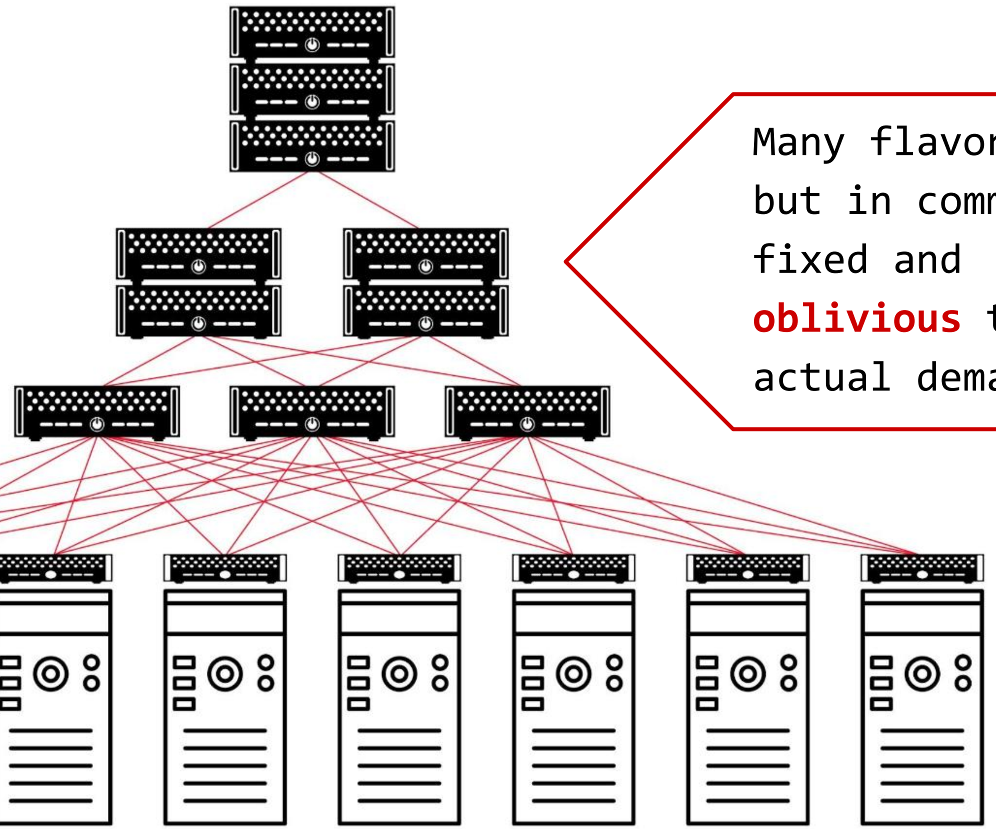


Many flavors,
but in common:
fixed and
oblivious to
actual demand.

Root Cause

Fixed and Demand-Oblivious Topology

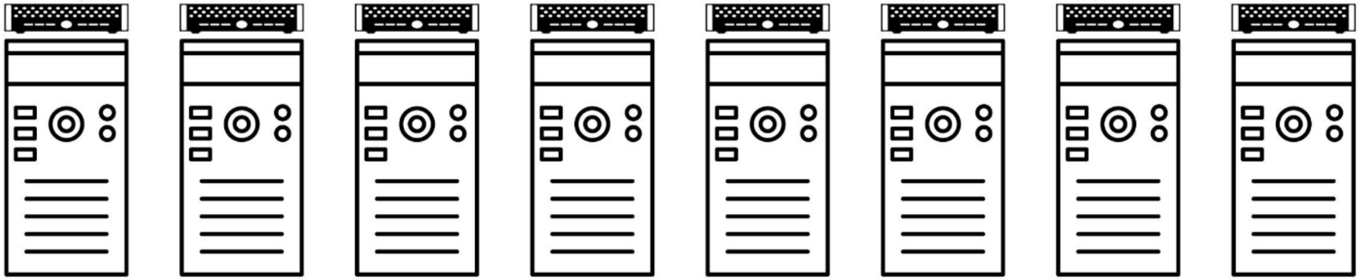
Highway which ignores actual traffic: frustrating!



Many flavors, but in common: fixed and **oblivious** to actual demand.

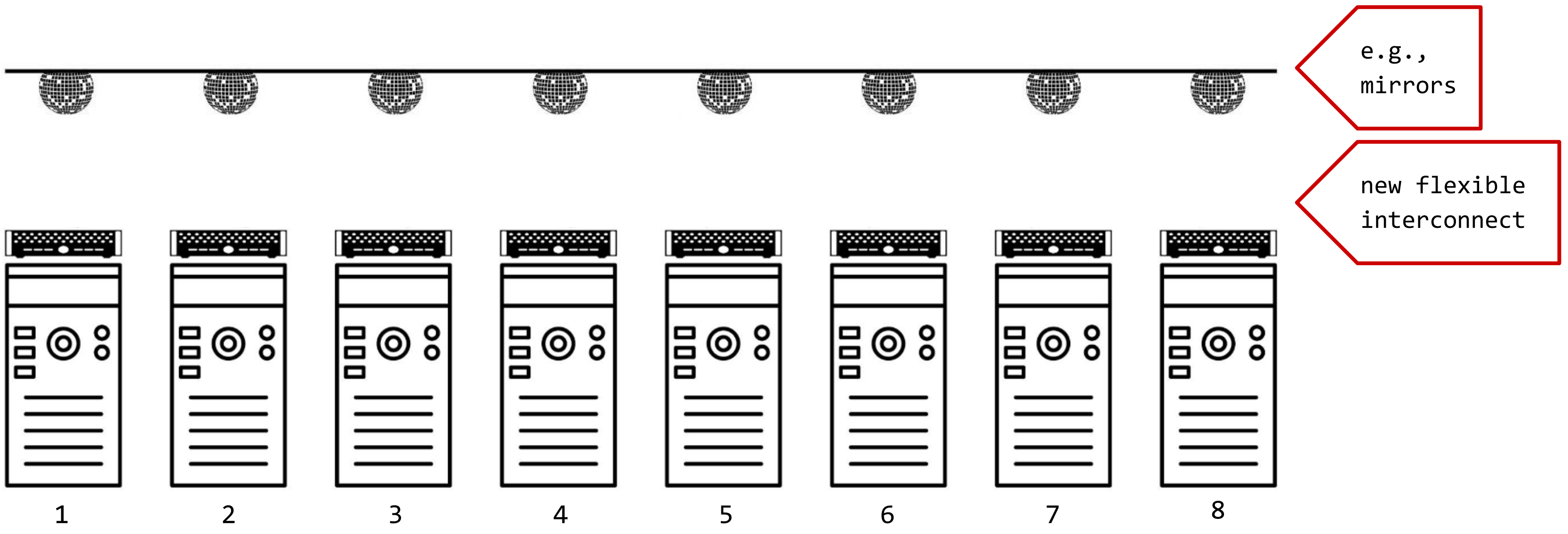
Our Vision

Flexible and Demand-Aware Topologies



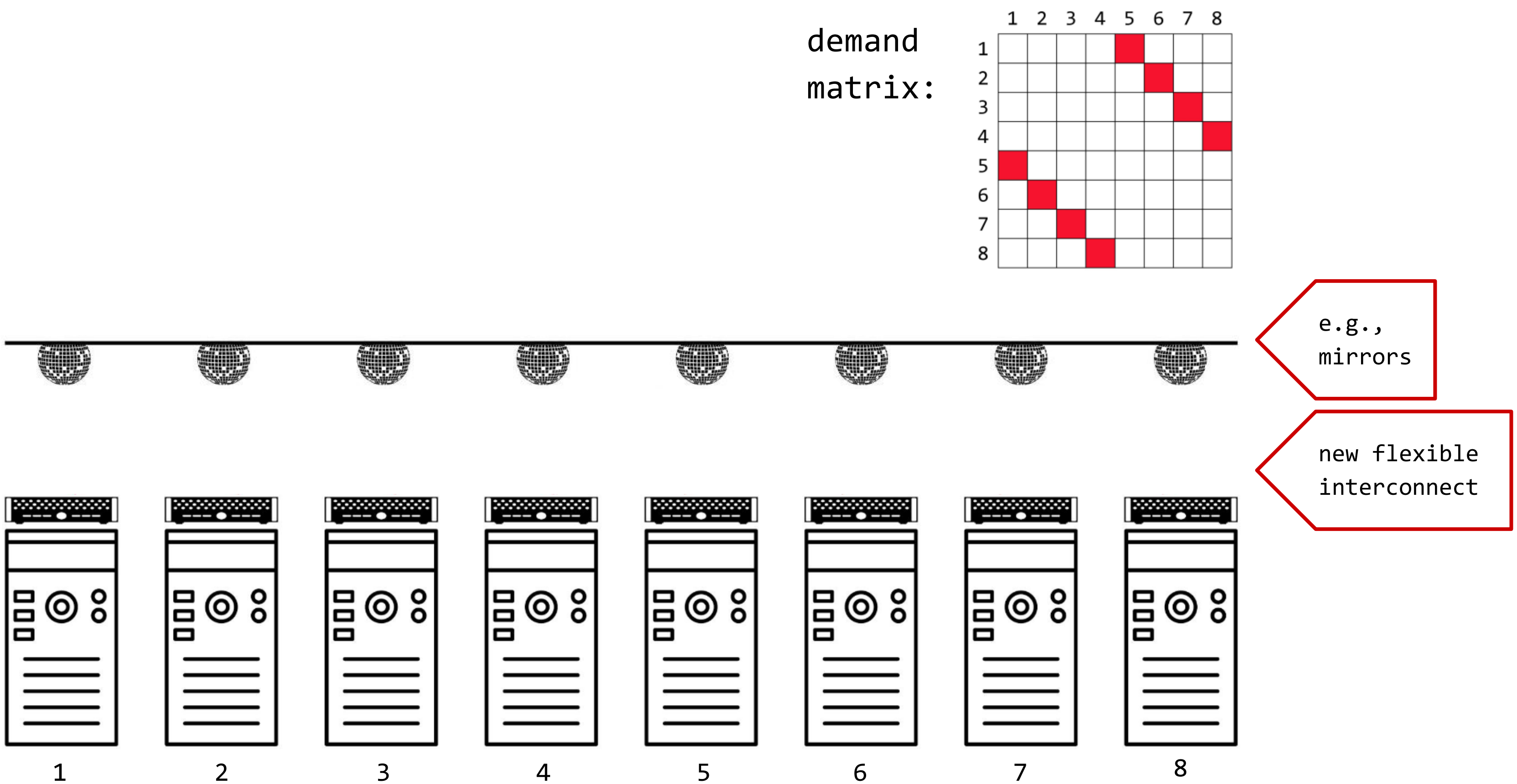
Our Vision

Flexible and Demand-Aware Topologies



Our Vision

Flexible and Demand-Aware Topologies

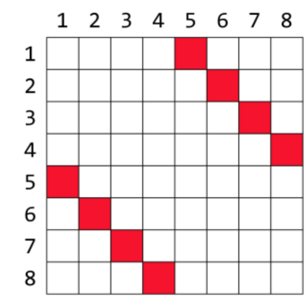


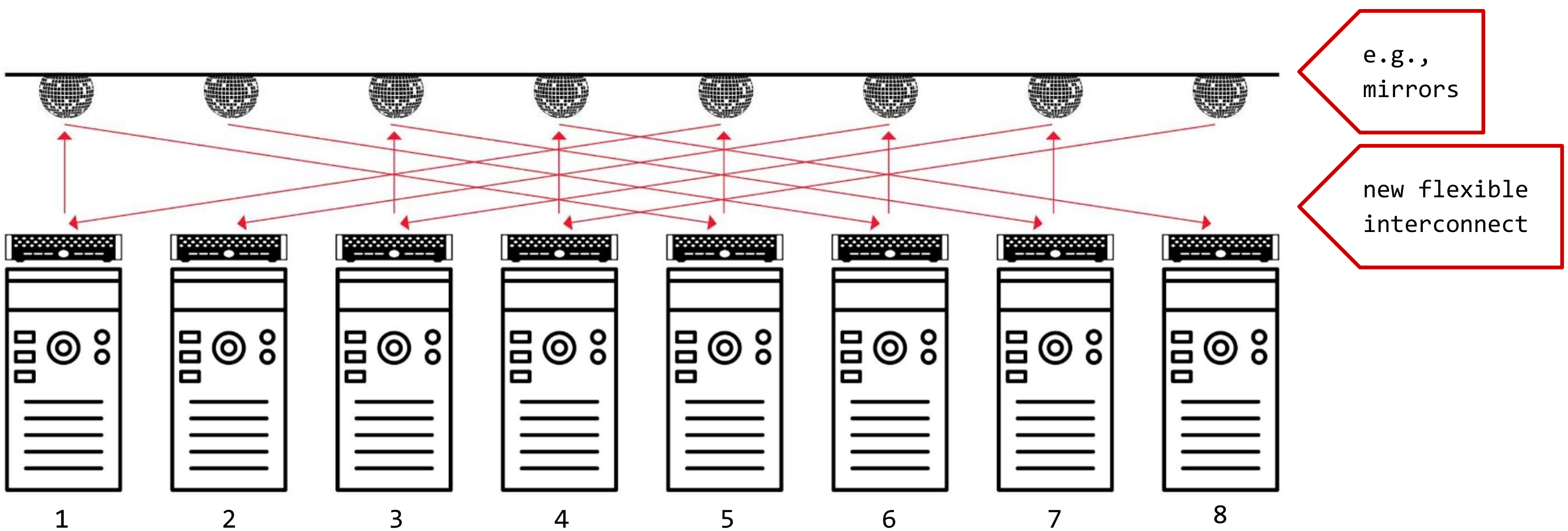
Our Vision

Flexible and Demand-Aware Topologies

Matches demand

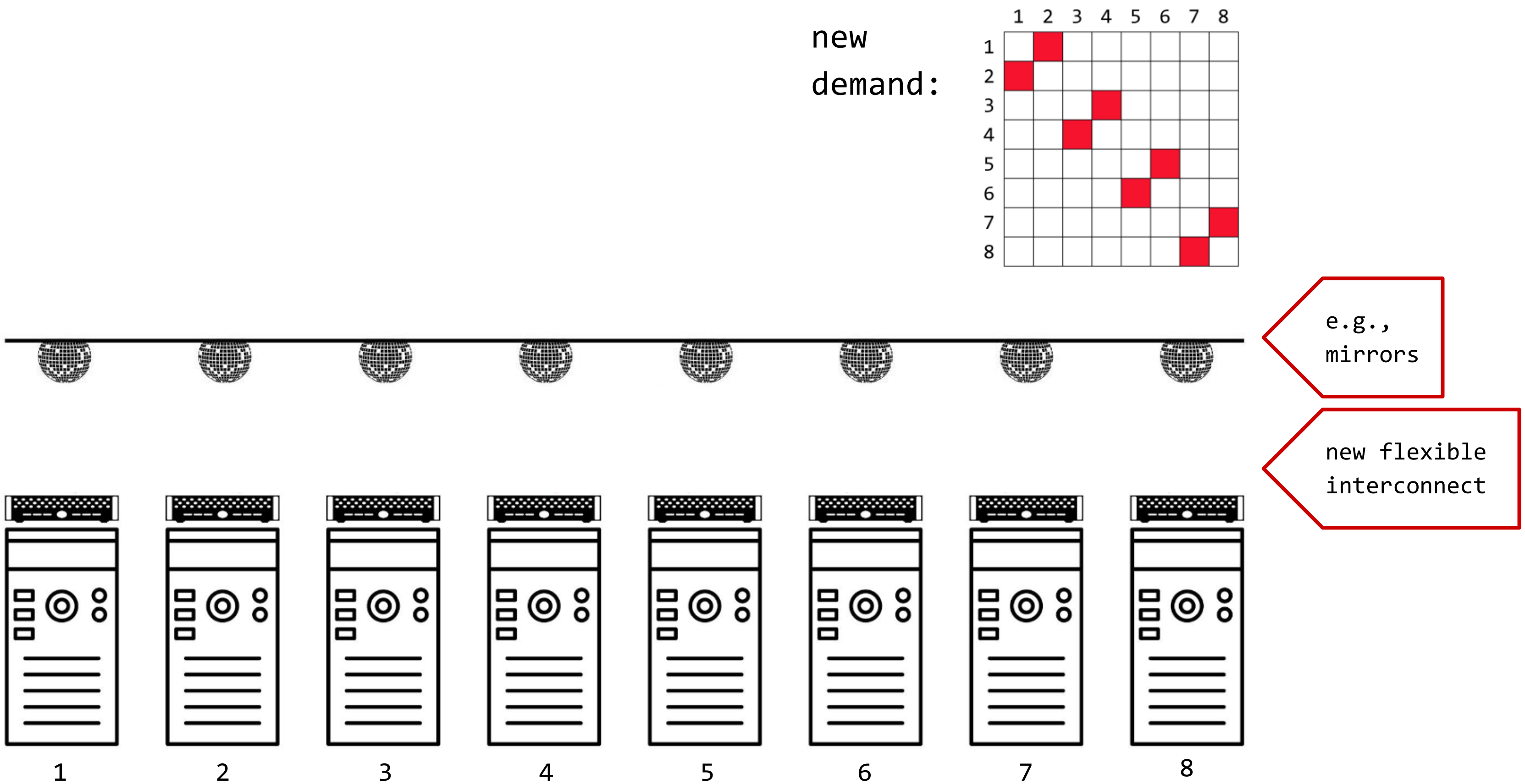
demand matrix:





Our Vision

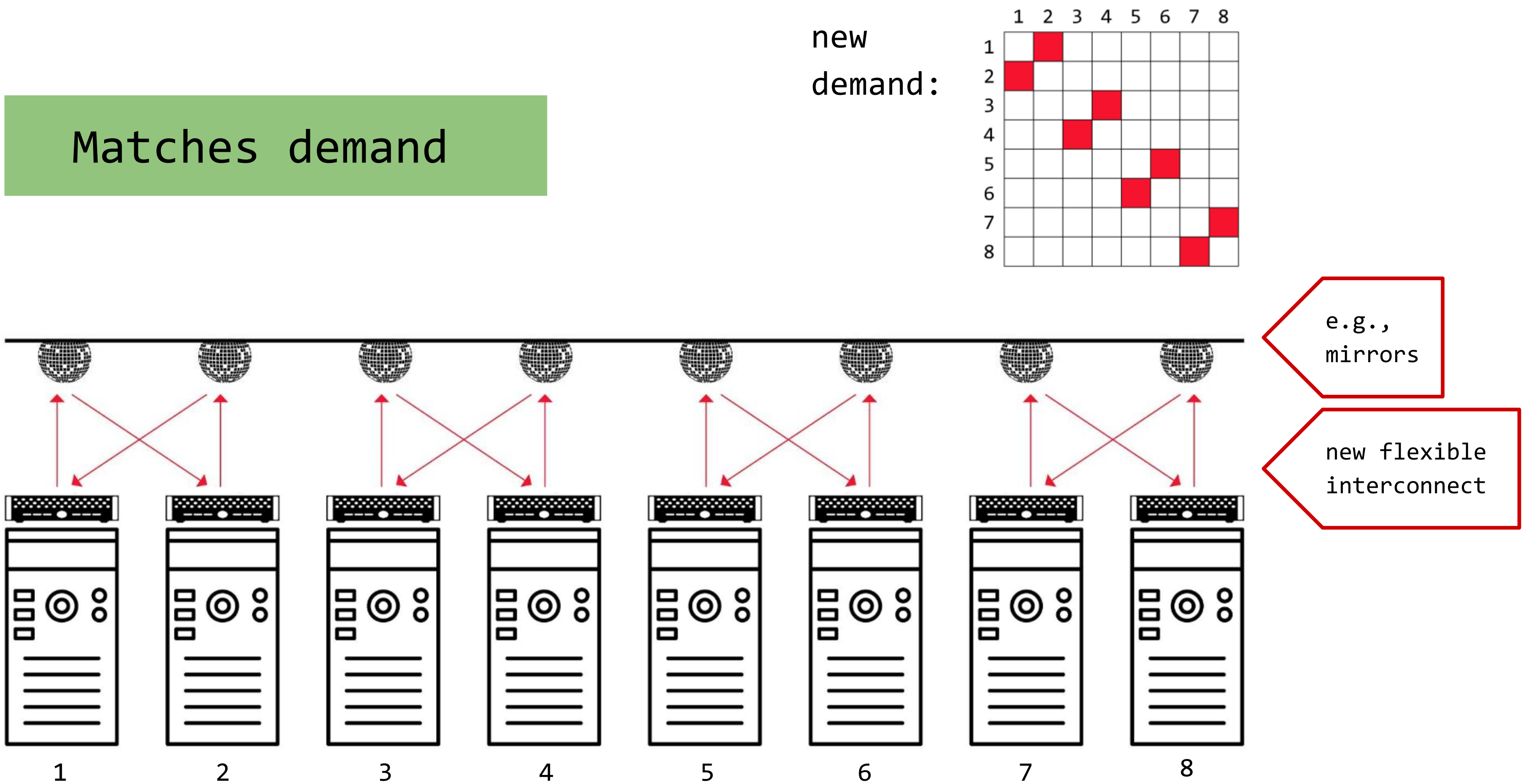
Flexible and Demand-Aware Topologies



Our Vision

Flexible and Demand-Aware Topologies

Matches demand

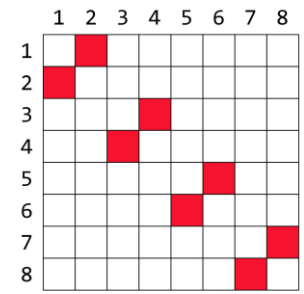


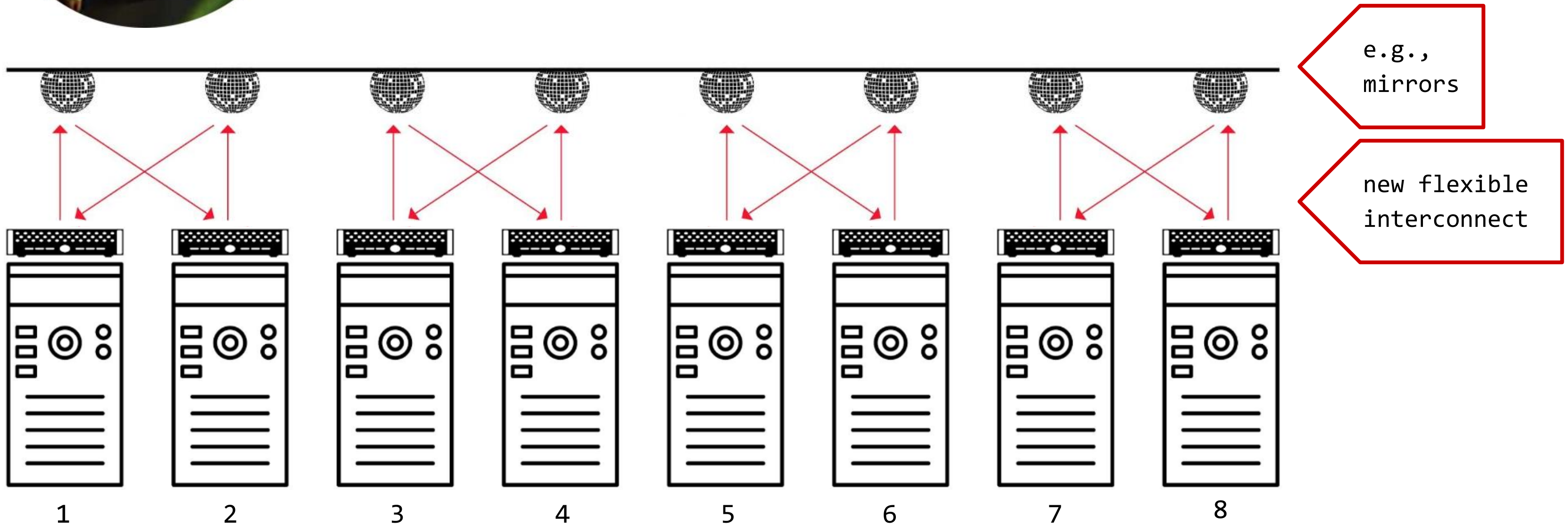
Our Vision

Flexible and Demand-Aware Topologies

Self-Adjusting Networks

new demand:



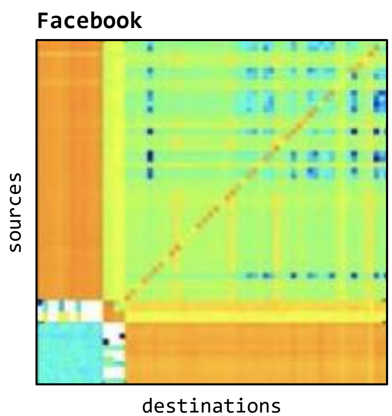


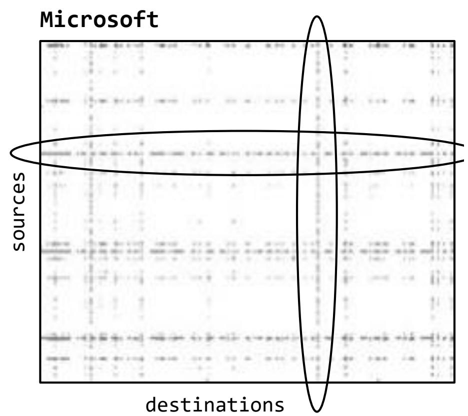
Our Motivation

Much Structure in the Demand

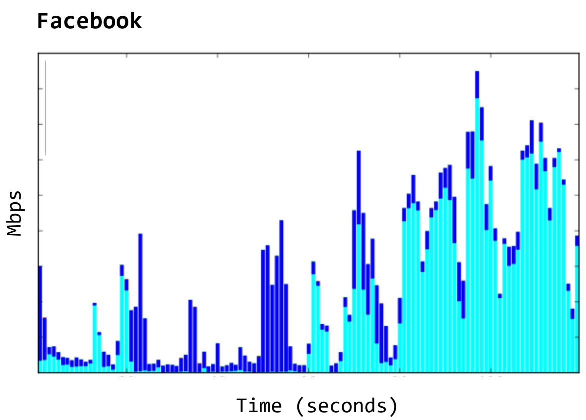
Empirical studies:

traffic matrices sparse and skewed



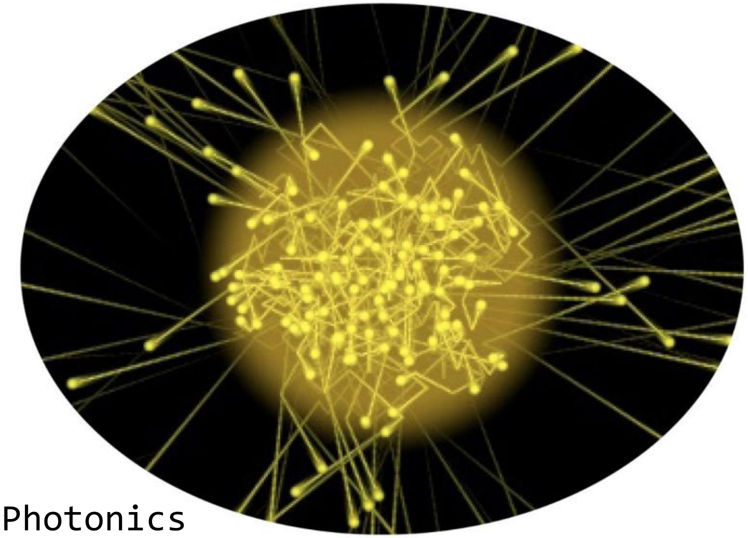


traffic bursty over time



My hypothesis: can be exploited.

Sounds Crazy? Emerging Enabling Technology.



Photonics

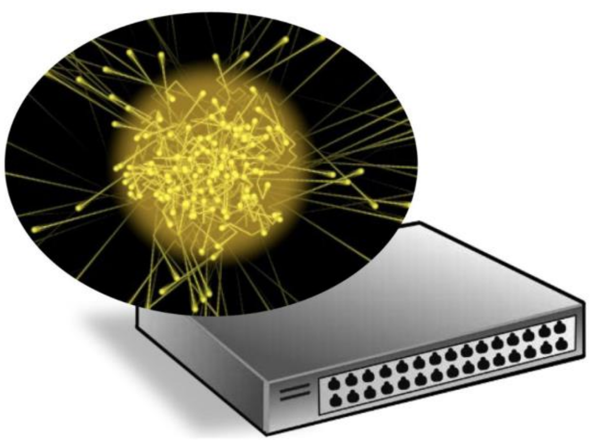
H2020:
“Photonics one of only five key enabling technologies for future prosperity.”

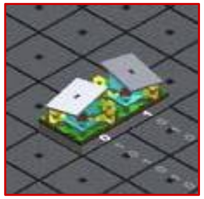
US National Research Council:
“Photons are the new Electrons.”

Enabler

Novel Reconfigurable Optical Switches

- **Spectrum** of prototypes
  - Different sizes, different reconfiguration times
  - From our last year's ACM **SIGCOMM** workshop OptSys





Prototype 1

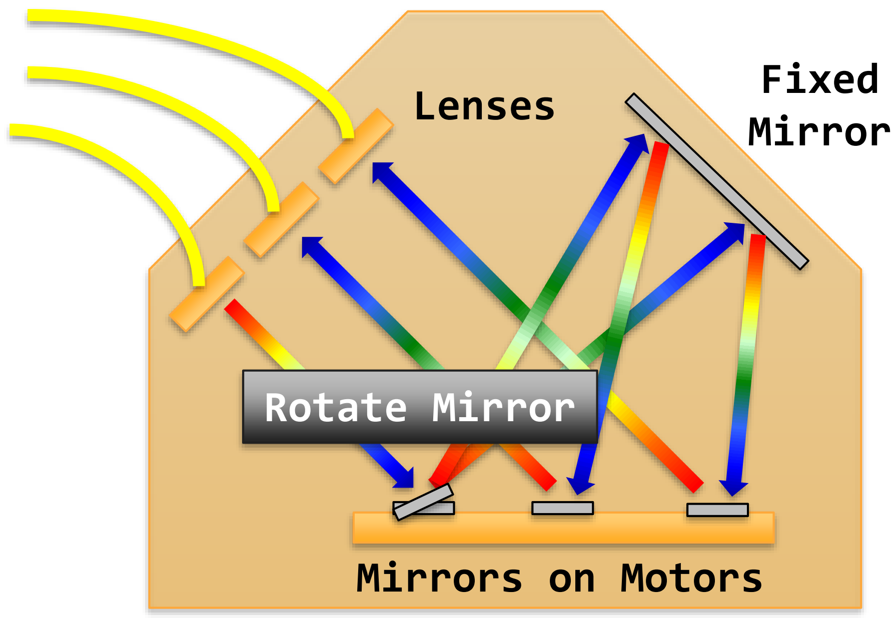
Prototype 2

Prototype 3

Example

Optical Circuit Switch

- ⇢ Optical Circuit Switch rapid adaption of physical layer
  - → Based on rotating mirrors



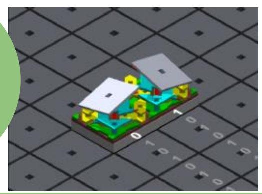
Optical Circuit Switch

By Nathan Farrington, SIGCOMM 2010

The Big Picture

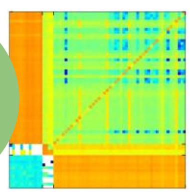
New!

Flexibility



More!

Structure



Self-Adjusting Networks

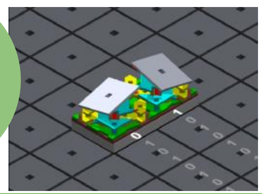
Efficiency

Now is the time!

The Big Picture

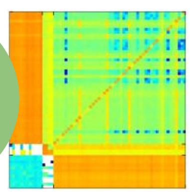
New!

Flexibility



More!

Structure



Self-Adjusting Networks

Now is the time!

Efficiency

Our goal: Develop the theoretical **foundations** of demand-aware, self-adjusting networks.

Unique Position

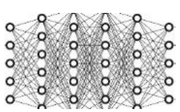
Demand-Aware, Self-Adjusting Systems

Everywhere, but mainly in software

Algorithmic trading

NETFLIX

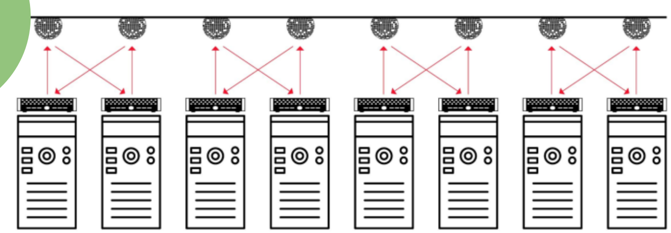
Recommender systems



Neural networks

VS

Our focus: in hardware



Question 1:

How to Quantify such “Structure” in the Demand?

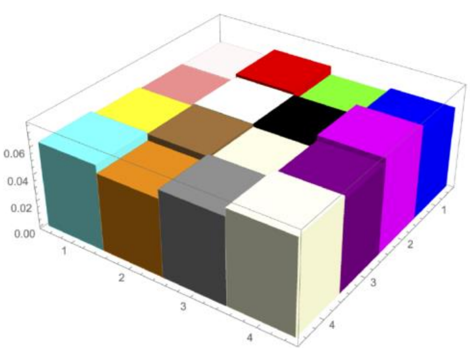
Intuition

Which demand has more structure?

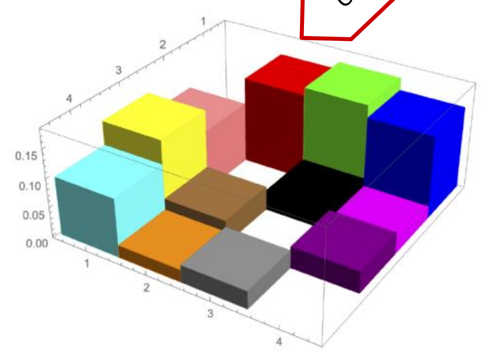
⇢ Traffic matrices of two different distributed ML applications

- ⇀ GPU-to-GPU

Color = communication pair



VS



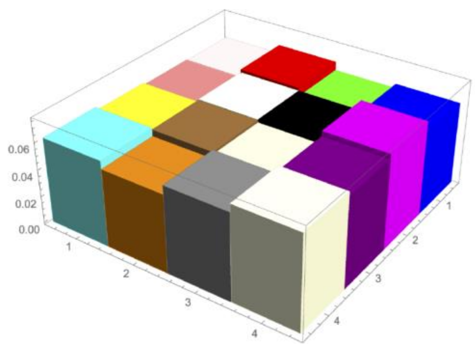
Intuition

Which demand has more structure?

⇢ Traffic matrices of two different distributed ML applications

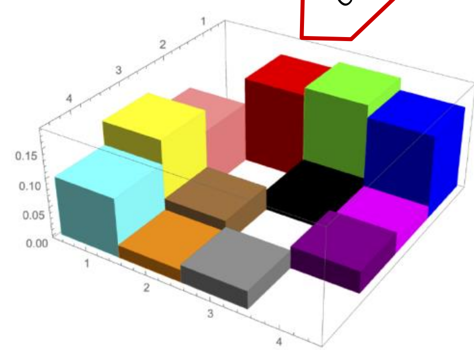
→ GPU-to-GPU

Color = communication pair



More uniform

VS

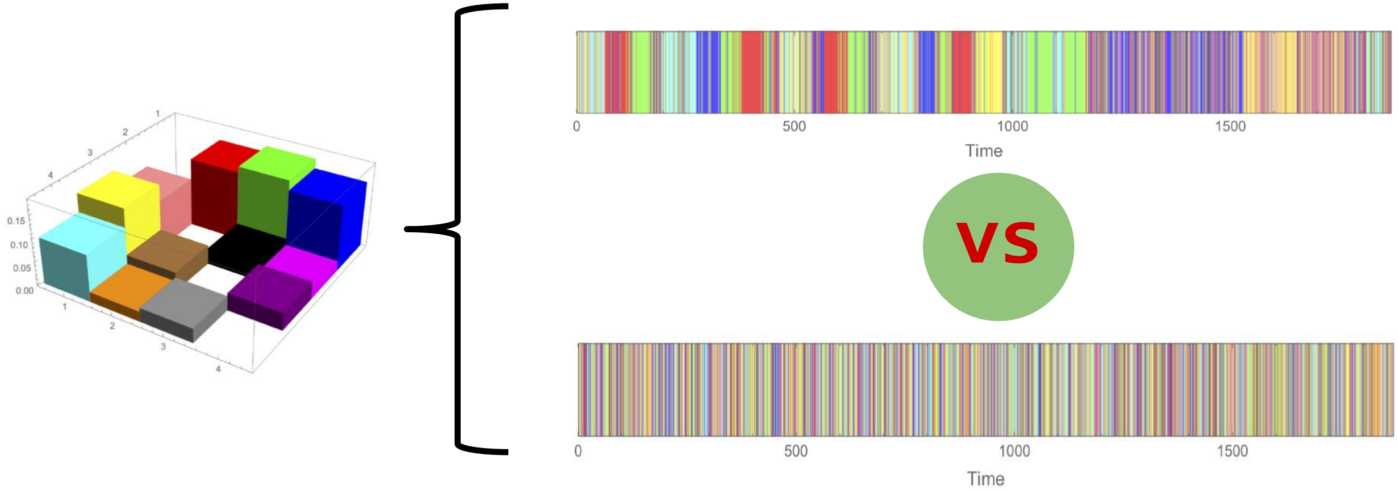


More structure

Intuition

Spatial vs temporal structure

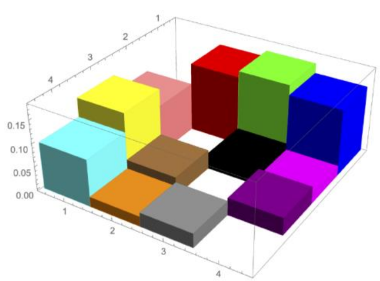
- Two different ways to generate same traffic matrix:
  - Same non-temporal structure
- Which one has more structure?

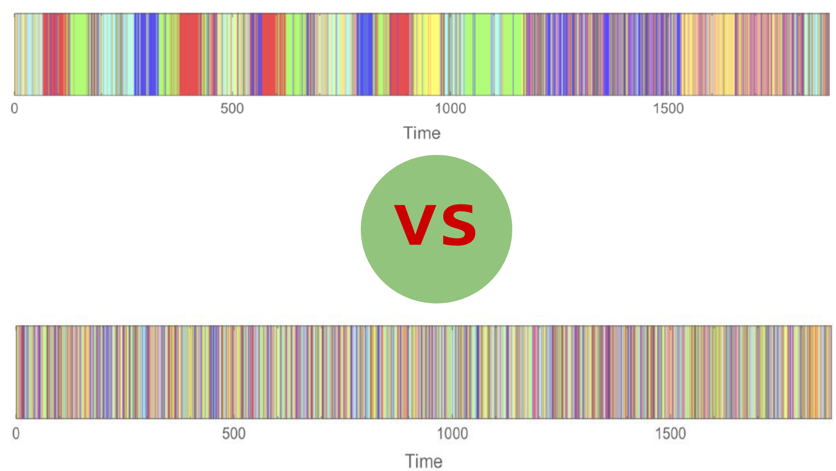


Intuition

Spatial vs temporal structure

- Two different ways to generate same traffic matrix:
  - Same non-temporal structure
- Which one has more structure?



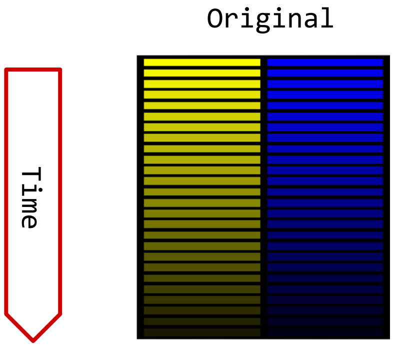


Systematically?

Trace Complexity

Information-Theoretic Approach

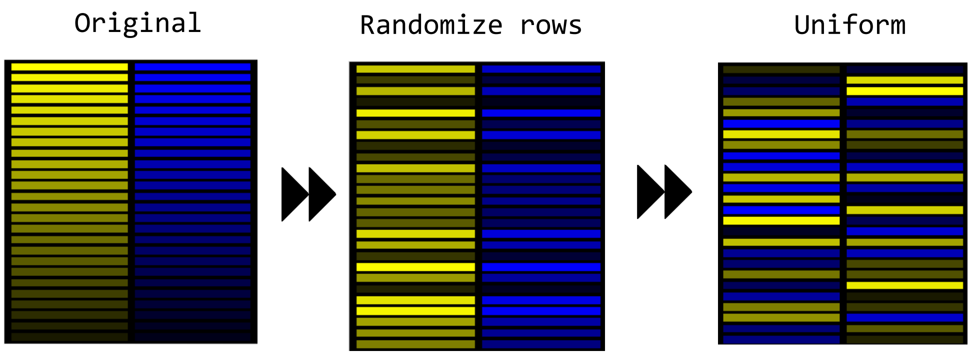
“Shuffle&Compress”



Trace Complexity

Information-Theoretic Approach

“Shuffle&Compress”



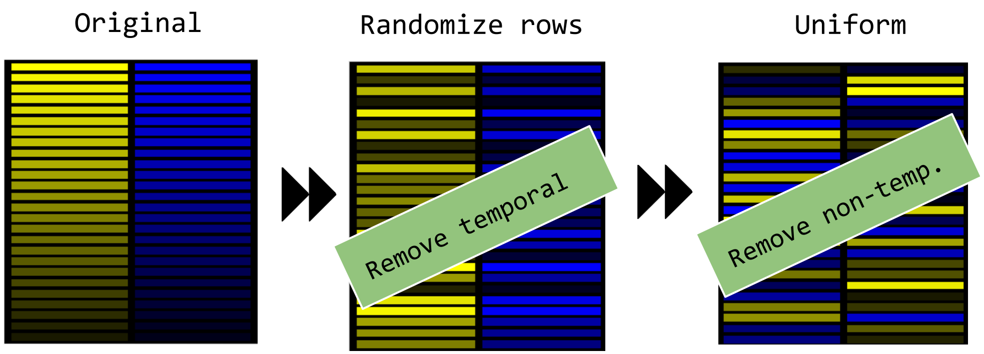
Increasing complexity (systematically randomized)

More structure (compresses better)

Trace Complexity

Information-Theoretic Approach

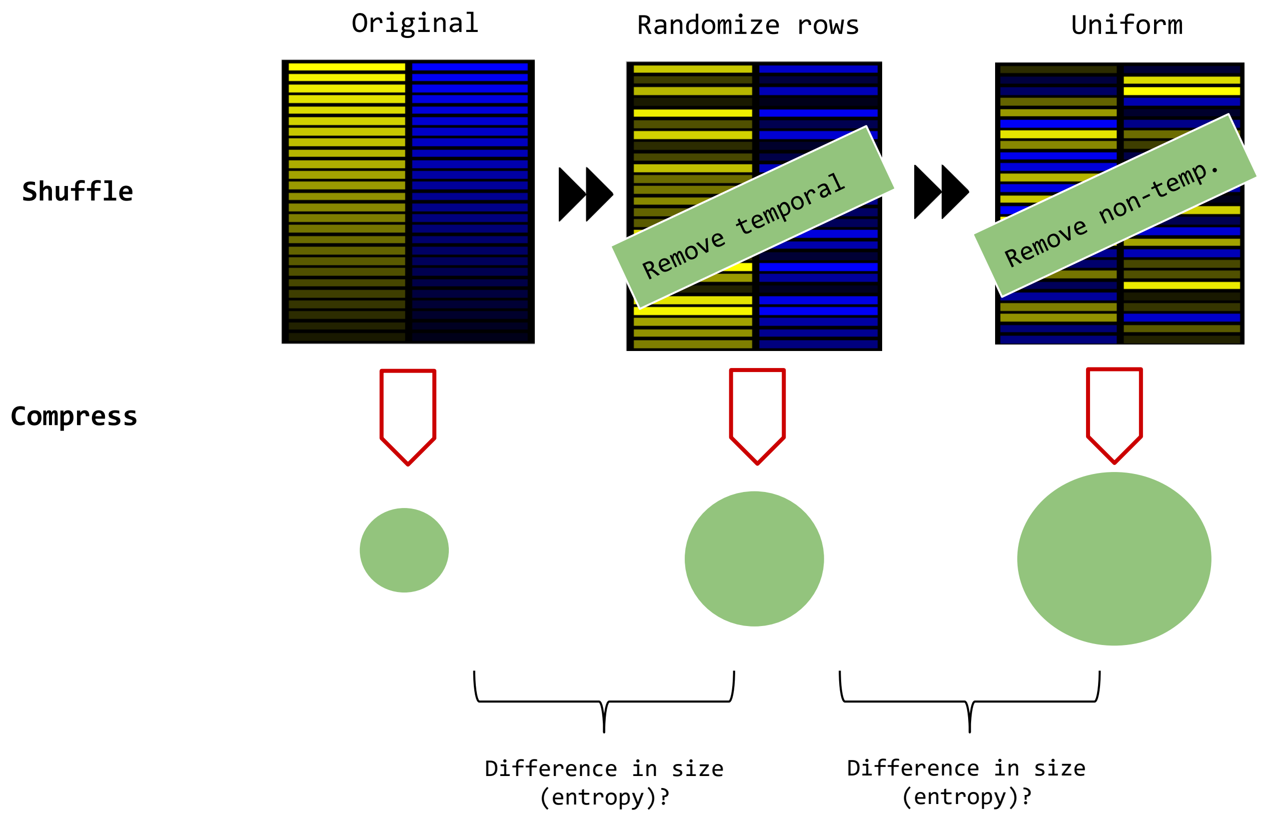
“Shuffle&Compress”



Trace Complexity

Information-Theoretic Approach

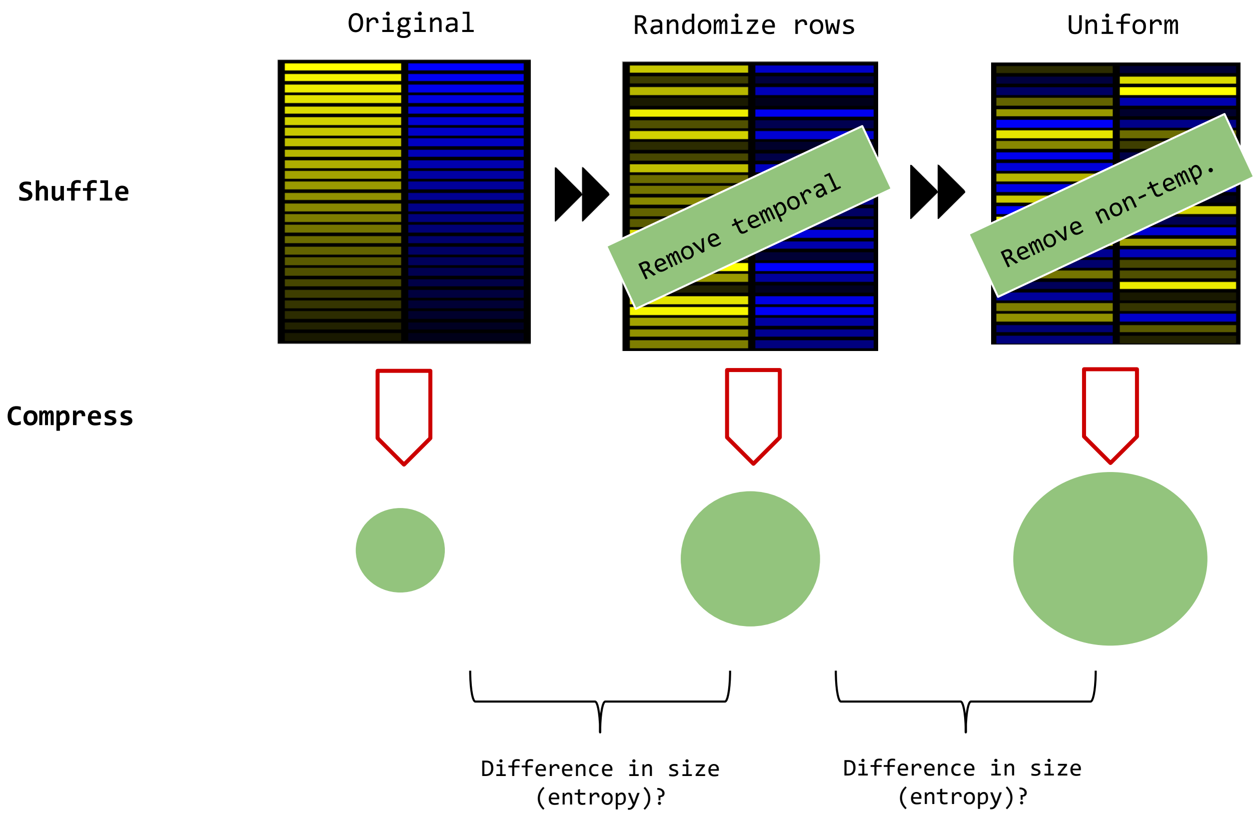
“Shuffle&Compress”



Trace Complexity

Information-Theoretic Approach

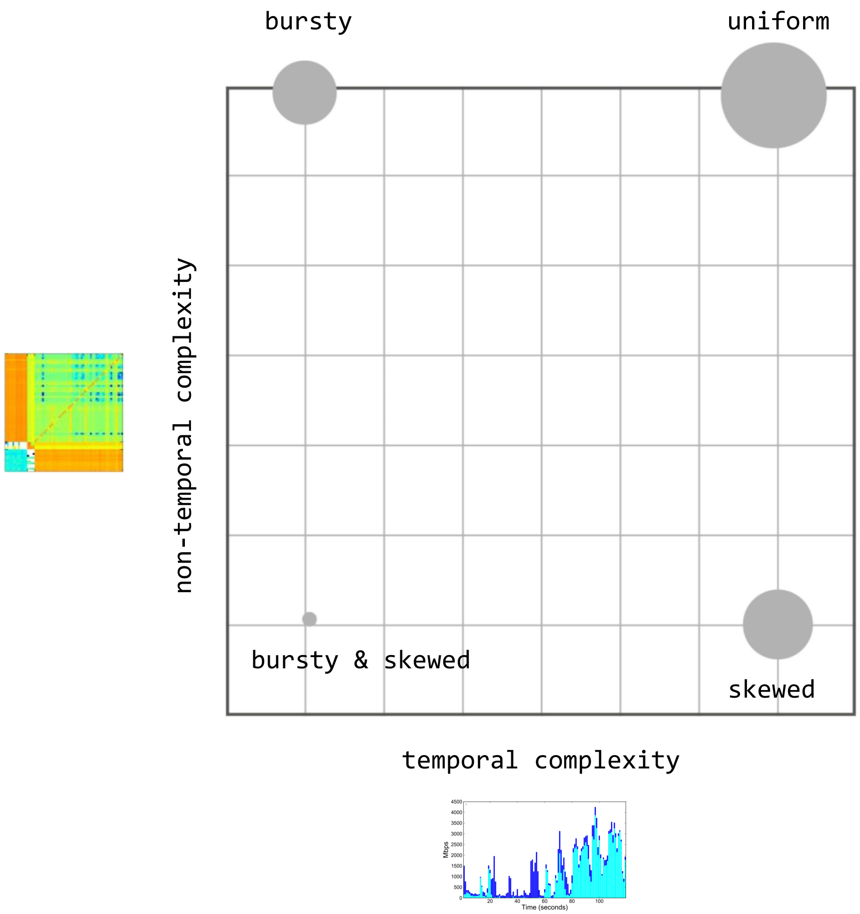
“Shuffle&Compress”



Can be used to define 2-dimensional complexity map!

Our Methodology

Complexity Map

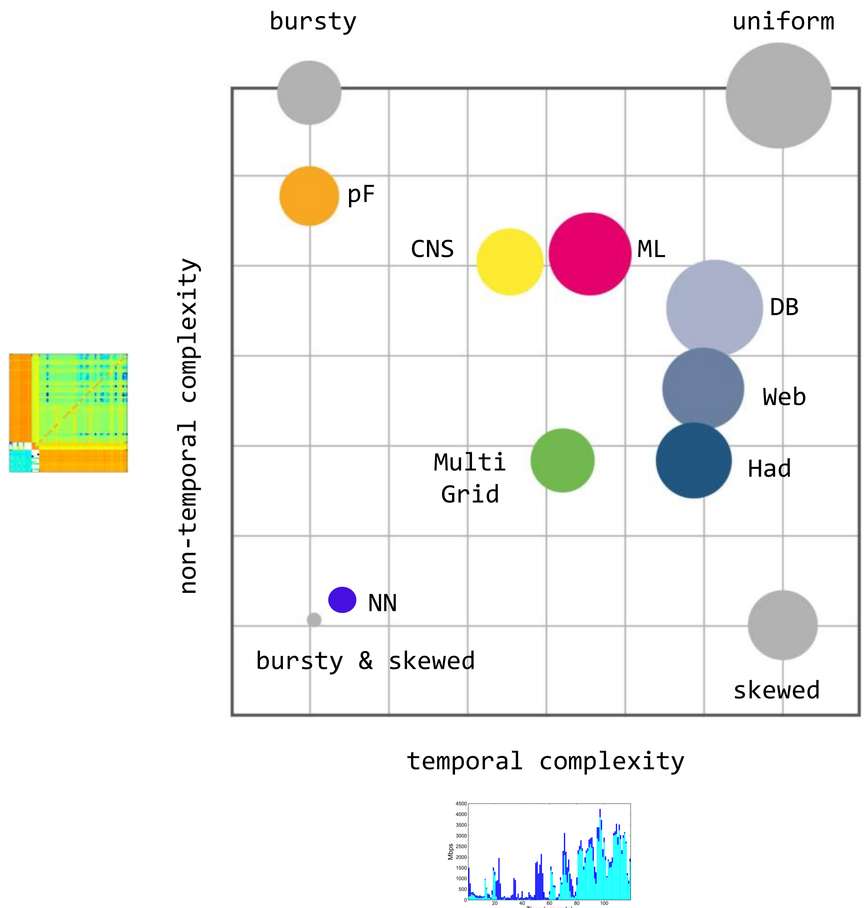


No structure

Our **approach:** iterative **randomization and compression** of trace to identify dimensions of structure.

Our Methodology

Complexity Map



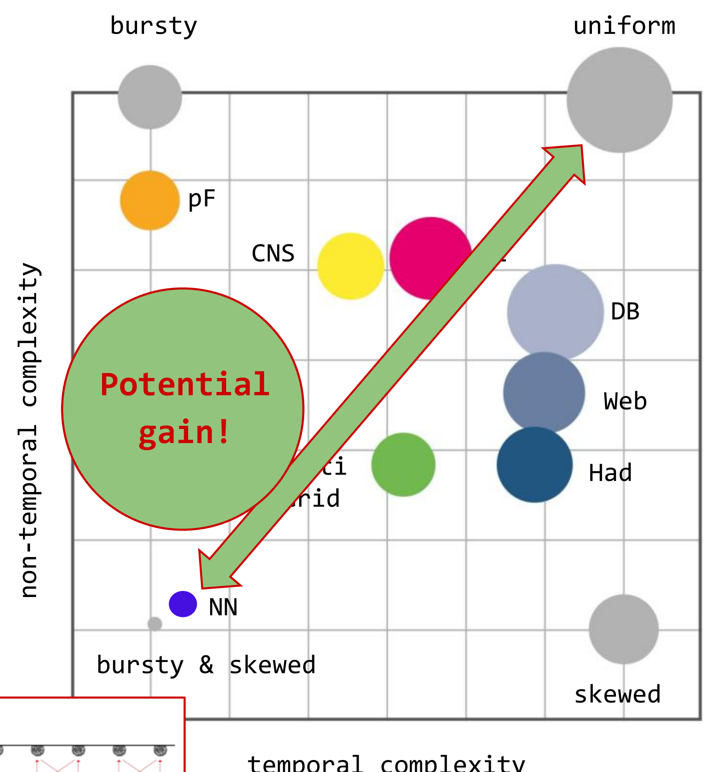
No structure

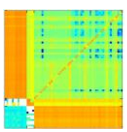
Our **approach**: iterative **randomization and compression** of trace to identify dimensions of structure.

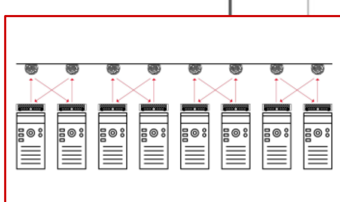
Different structures!

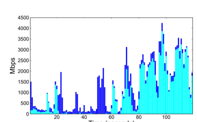
Our Methodology

Complexity Map









Our **approach**: iterative **randomization and compression** of trace to identify dimensions of structure.

Different structures!

Further Reading

ACM SIGMETRICS 2020

On the Complexity of Traffic Traces and Implications

CHEN AVIN, School of Electrical and Computer Engineering, Ben Gurion University of the Negev, Israel
MANYA GHOBADI, Computer Science and Artificial Intelligence Laboratory, MIT, USA
CHEN GRINER, School of Electrical and Computer Engineering, Ben Gurion University of the Negev, Israel
STEFAN SCHMID, Faculty of Computer Science, University of Vienna, Austria

This paper presents a systematic approach to identify and quantify the types of structures featured by packet traces in communication networks. Our approach leverages an information-theoretic methodology, based on iterative randomization and compression of the packet trace, which allows us to systematically remove and measure dimensions of structure in the trace. In particular, we introduce the notion of *trace complexity* which approximates the entropy rate of a packet trace. Considering several real-world traces, we show that trace complexity can provide unique insights into the characteristics of various applications. Based on our approach, we also propose a traffic generator model able to produce a synthetic trace that matches the complexity levels of its corresponding real-world trace. Using a case study in the context of datacenters, we show that insights into the structure of packet traces can lead to improved demand-aware network designs: datacenter topologies that are optimized for specific traffic patterns.

CCS Concepts: • **Networks** → **Network performance evaluation**; **Network algorithms**; **Data center networks**; • **Mathematics of computing** → *Information theory*;

Additional Key Words and Phrases: trace complexity, self-adjusting networks, entropy rate, compress, complexity map, data centers

20

ACM Reference Format:
Chen Avin, Manya Ghobadi, Chen Griner, and Stefan Schmid. 2020. On the Complexity of Traffic Traces and Implications. *Proc. ACM Meas. Anal. Comput. Syst.* 4, 1, Article 20 (March 2020), 29 pages. https://doi.org/10.1145/3379486

1 INTRODUCTION

Packet traces collected from networking applications, such as datacenter traffic, have been shown to feature much *structure*: datacenter traffic matrices are sparse and skewed [16, 39], exhibit

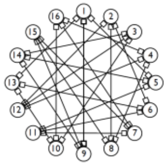
Question 2:

Given This Structure, What Can Be Achieved? Metrics and Algorithms?

A first insight: entropy of the demand.

Models and Connection to Datastructures & Coding

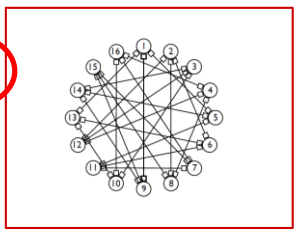
Oblivious networks
(worst-case traffic)



More structure: **lower routing cost**

Models and Connection to Datastructures & Coding

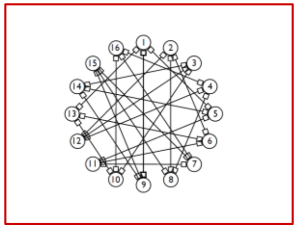
Oblivious networks
(worst-case traffic)



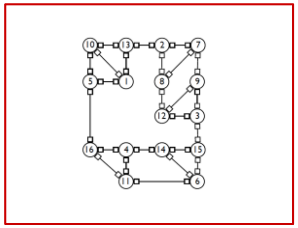
More structure: **lower routing cost**

Models and Connection to Datastructures & Coding

Oblivious networks
(worst-case traffic)



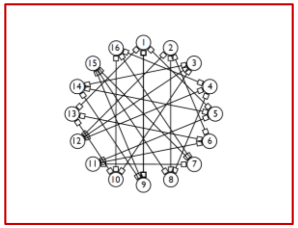
Demand-aware networks
(spatial structure)



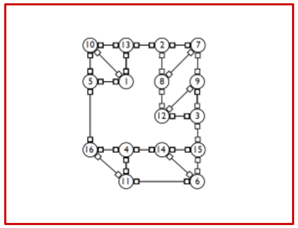
More structure: **lower routing cost**

Models and Connection to Datastructures & Coding

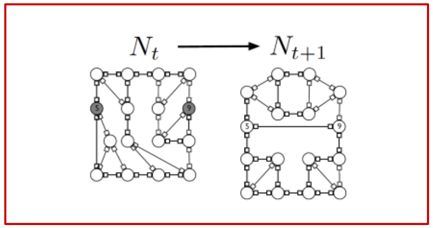
Oblivious networks
(worst-case traffic)



Demand-aware networks
(spatial structure)



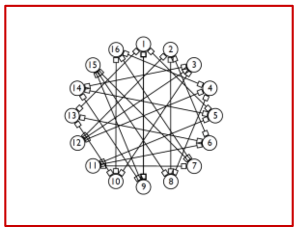
Self-adjusting networks
(temporal structure)



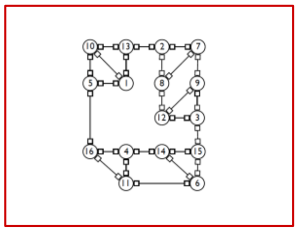
More structure: **lower routing cost**

Models and Connection to Datastructures & Coding

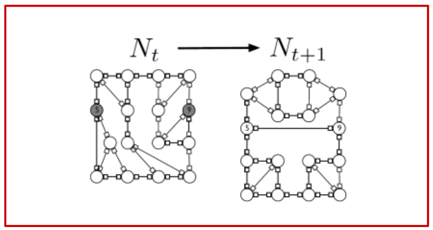
Oblivious networks
(worst-case traffic)



Demand-aware networks
(spatial structure)

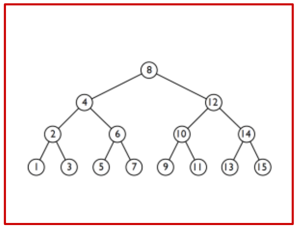


Self-adjusting networks
(temporal structure)

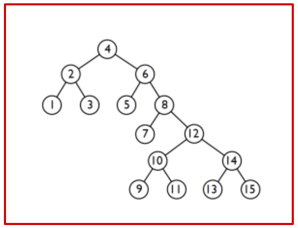


More structure: **lower routing cost**

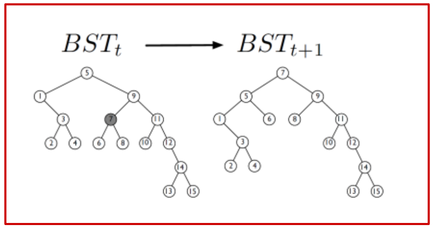
Traditional BST
(Worst-case coding)



Demand-aware BST
(Huffman coding)



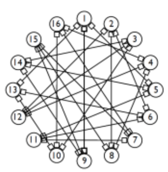
Self-adjusting BST
(Dynamic Huffman coding)



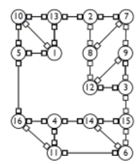
More structure: improved **access cost** / shorter **codes**

Models and Connection to Datastructures & Coding

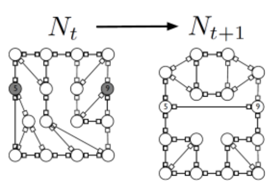
Oblivious networks
(worst-case traffic)



Demand-aware networks
(spatial structure)



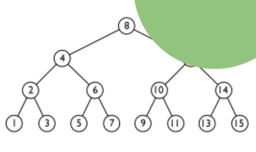
Self-adjusting networks
(temporal structure)



More structure: **lower routing cost**

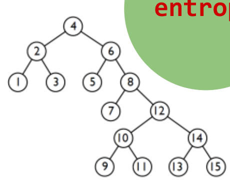
Traditional BST
(Worst-case

log n



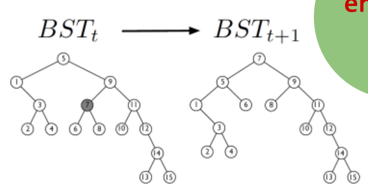
Demand-aware BST
(Huffman c

entropy



Self-adjusting BST
(Dynamic Huffman codin

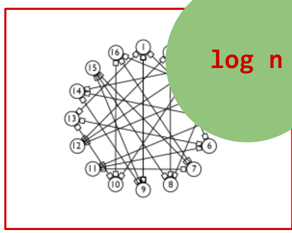
entropy rate



More structure: improved **access cost** / shorter **codes**

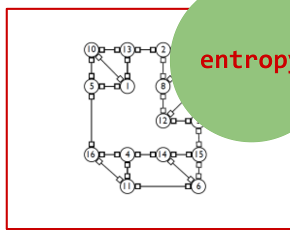
Models and Connection to Datastructures & Coding

Traditional networks
(worst-case traffic)



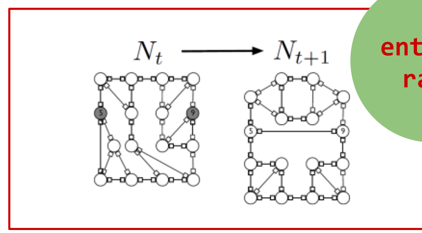
log n

Demand-aware networks
(spatial structure)



entropy

Self-adjusting networks
(temporal structure)

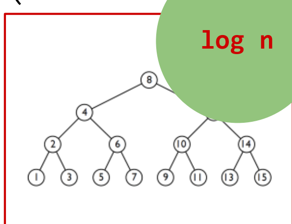


entropy rate

More than an analogy!

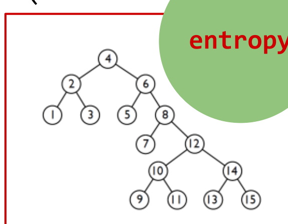
More structur lower routing cost

Traditional BST
(Worst-case



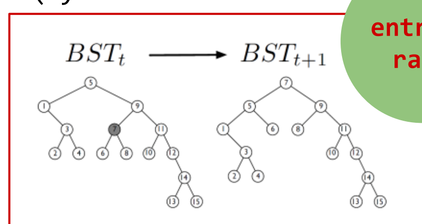
log n

Demand-aware BST
(Huffman c



entropy

Self-adjusting BST
(Dynamic Huffman codin



entropy rate

More structure: improved **access cost** / shorter **codes**

Generalize methodology:
... and transfer entropy bounds and algorithms of data-structures to networks.

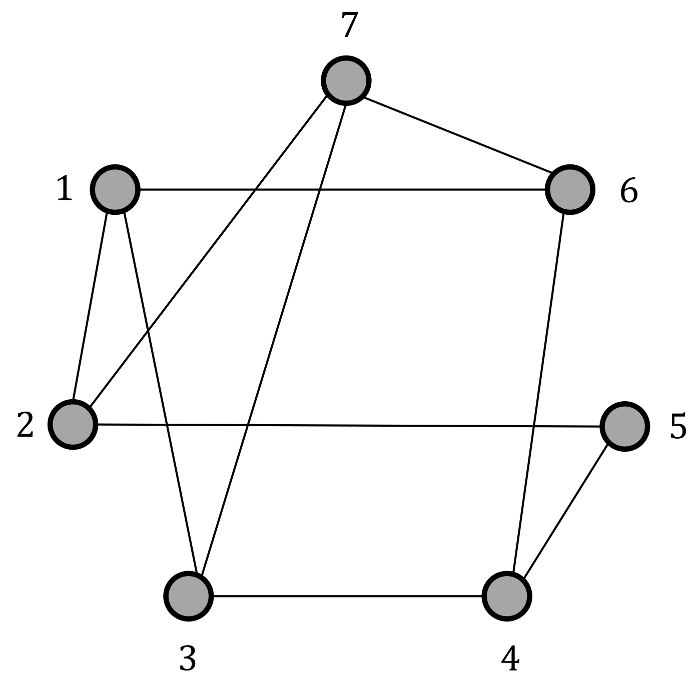
First result:
Demand-aware networks of asymptotically optimal route lengths.

Case Study “Route Lengths”

Constant-Degree Demand-Aware Network

Destinations

Sources	1	2	3	4	5	6	7
1	0	$\frac{2}{65}$	$\frac{1}{13}$	$\frac{1}{65}$	$\frac{1}{65}$	$\frac{2}{65}$	$\frac{3}{65}$
2	$\frac{2}{65}$	0	$\frac{1}{65}$	0	0	0	$\frac{2}{65}$
3	$\frac{1}{13}$	$\frac{1}{65}$	0	$\frac{2}{65}$	0	0	$\frac{1}{13}$
4	$\frac{1}{65}$	0	$\frac{2}{65}$	0	$\frac{4}{65}$	0	0
5	$\frac{1}{65}$	0	$\frac{3}{65}$	$\frac{4}{65}$	0	0	0
6	$\frac{2}{65}$	0	0	0	0	0	$\frac{3}{65}$
7	$\frac{3}{65}$	$\frac{2}{65}$	$\frac{1}{13}$	0	0	$\frac{3}{65}$	0



$$\mathrm{ERL}(\mathcal{D},\mathrm{N}) = \sum_{(u,v)\in\mathcal{D}} p(u,v)\cdot d_N(u,v)$$

Case Study "Route Lengths"

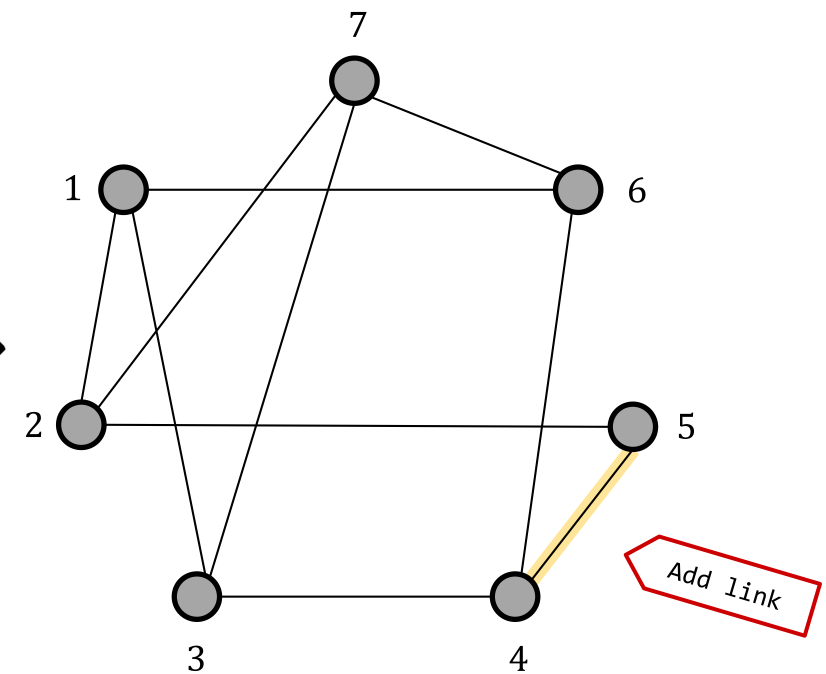
Constant-Degree Demand-Aware Network

Destinations

Sources

	1	2	3	4	5	6	7
1	0	$\frac{2}{65}$	$\frac{1}{13}$	$\frac{1}{65}$	$\frac{1}{65}$	$\frac{2}{65}$	$\frac{3}{65}$
2	$\frac{2}{65}$	0	$\frac{1}{65}$	0	0	0	$\frac{2}{65}$
3	$\frac{1}{13}$	$\frac{1}{65}$	0	$\frac{2}{65}$	0	0	$\frac{1}{13}$
4	$\frac{1}{65}$	0	$\frac{2}{65}$	0	$\frac{4}{65}$	0	0
5	$\frac{1}{65}$	0	$\frac{3}{65}$	[illegible]	0	0	0
6	$\frac{2}{65}$	0	[illegible]	0	0	0	$\frac{3}{65}$
7	$\frac{3}{65}$	[illegible]	$\frac{1}{13}$	0	0	$\frac{3}{65}$	0

Much from 4 to 5



$$\mathrm{ERL}(\mathcal{D},N) = \sum_{(u,v)\in\mathcal{D}} p(u,v) \cdot d_N(u,v)$$

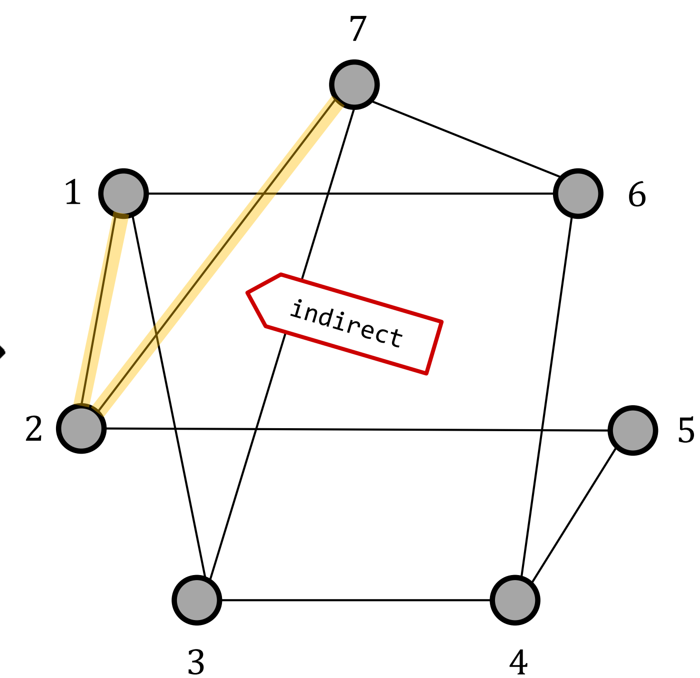
Case Study “Route Lengths”

Constant-Degree Demand-Aware Network

Communicated with many

Destinations

Sources	1	2	3	4	5	6	7
1	0	$\frac{2}{65}$	$\frac{1}{13}$	$\frac{1}{65}$	$\frac{1}{65}$	$\frac{2}{65}$	$\frac{3}{65}$
2	$\frac{2}{65}$	0	$\frac{1}{65}$	0	0	0	$\frac{2}{65}$
3	$\frac{1}{13}$	$\frac{1}{65}$	0	$\frac{2}{65}$	0	0	$\frac{1}{13}$
4	$\frac{1}{65}$	0	$\frac{2}{65}$	0	$\frac{4}{65}$	0	0
5	$\frac{1}{65}$	0	$\frac{3}{65}$	$\frac{4}{65}$	0	0	0
6	$\frac{2}{65}$	0	0	0	0	0	$\frac{3}{65}$
7	$\frac{3}{65}$	$\frac{2}{65}$	$\frac{1}{13}$	0	0	$\frac{3}{65}$	0



$$\mathrm{ERL}(\mathcal{D},\mathrm{N}) = \sum_{(u,v)\in\mathcal{D}} p(u,v)\cdot d_N(u,v)$$

Case Study “Route Lengths”

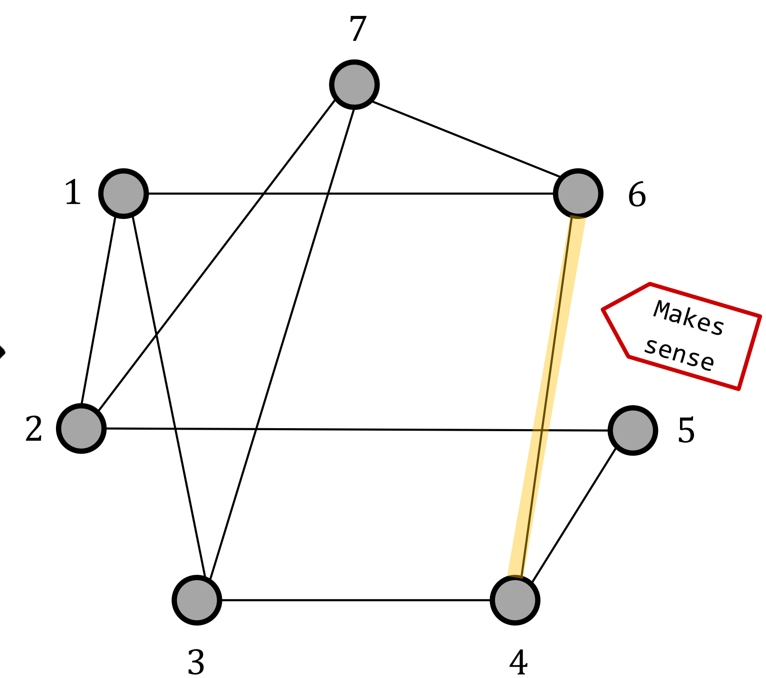
Constant-Degree Demand-Aware Network

Destinations

Sources

	1	2	3	4	5	6	7
1	0	$\frac{2}{65}$	$\frac{1}{13}$	$\frac{1}{65}$	$\frac{1}{65}$	$\frac{2}{65}$	$\frac{3}{65}$
2	$\frac{2}{65}$	0	[illegible]	[illegible]	0	0	$\frac{2}{65}$
3	$\frac{1}{13}$	$\frac{1}{65}$	0	[illegible]	[illegible]	0	$\frac{1}{13}$
4	$\frac{1}{65}$	0	$\frac{2}{65}$	0	[illegible]	0	0
5	$\frac{1}{65}$	0	$\frac{3}{65}$	$\frac{4}{65}$	0	0	0
6	$\frac{2}{65}$	0	0	0	0	0	$\frac{3}{65}$
7	$\frac{3}{65}$	$\frac{2}{65}$	$\frac{1}{13}$	0	0	$\frac{3}{65}$	0

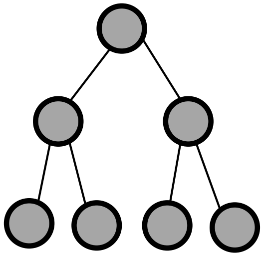
Don’t communicate



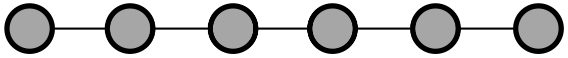
$$\mathrm{ERL}(\mathcal{D},N) = \sum_{(u,v)\in\mathcal{D}} p(u,v)\cdot d_N(u,v)$$

Examples

- DAN for △=3
  - E.g., complete **binary tree** would be **log n**
  - Can we do better?

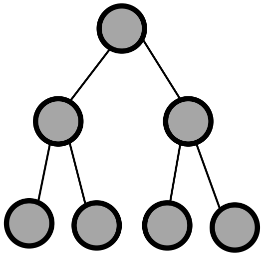


- DAN for △=2
  - Set of **lines** and **cycles**

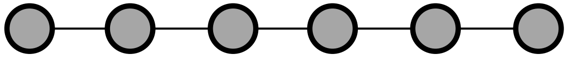


Examples

- DAN for △=3
  - E.g., complete **binary tree** would be **log n**
  - Can we do better?



- DAN for △=2
  - Set of **lines** and **cycles**



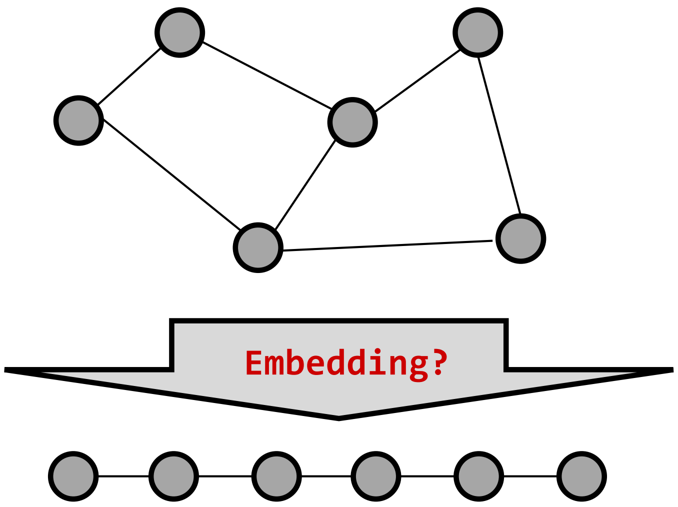
How hard?

Related Problem

Virtual Network Embedding Problem (VNEP)

Example △=2: A Minium Linear Arrangement (**MLA**) Problem

→ Minimizes sum of virtual edges

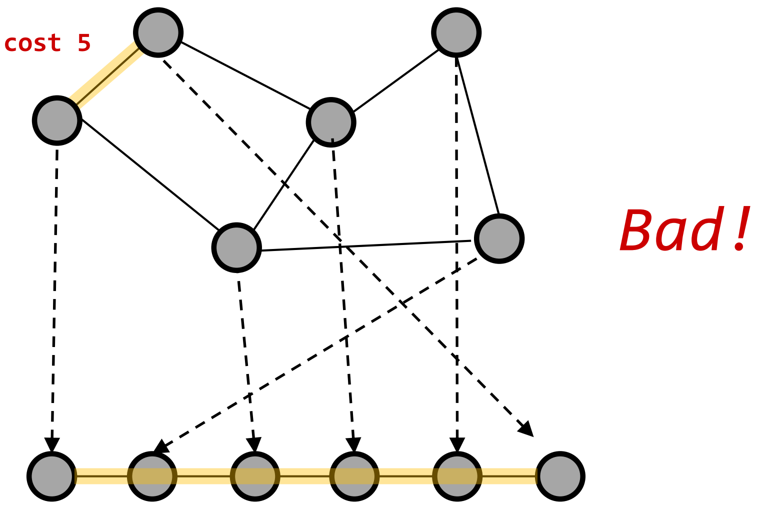


Related Problem

Virtual Network Embedding Problem (VNEP)

Example △=2: A Minium Linear Arrangement (**MLA**) Problem

→ Minimizes sum of virtual edges

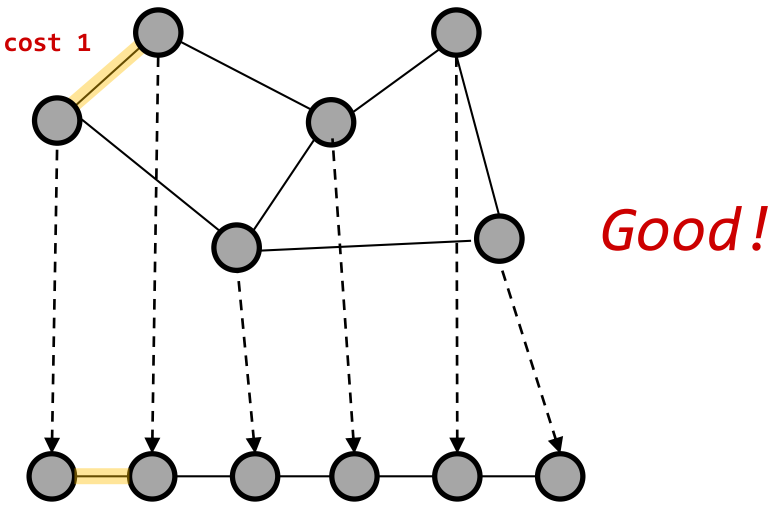


Related Problem

Virtual Network Embedding Problem (VNEP)

Example △=2: A Minium Linear Arrangement (**MLA**) Problem

→ Minimizes sum of virtual edges



Related Problem

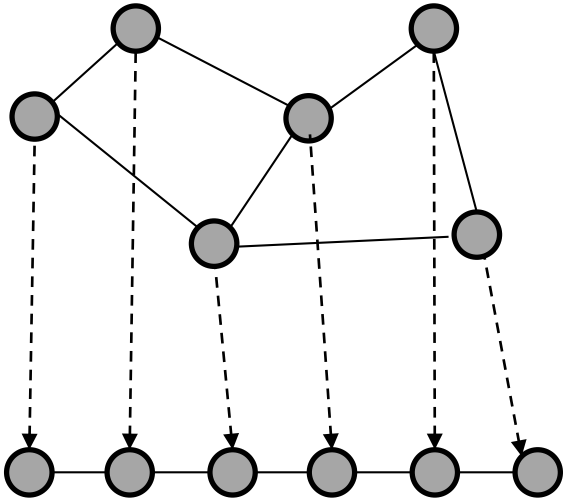
Virtual Network Embedding Problem (VNEP)

Example △=2: A Minium Linear Arrangement (**MLA**) Problem

→ Minimizes sum of virtual edges

MLA is **NP-hard**

→ … and so is our problem!



Related Problem

Virtual Network Embedding Problem (VNEP)

Example △=2: A Minium Linear Arrangement (**MLA**) Problem

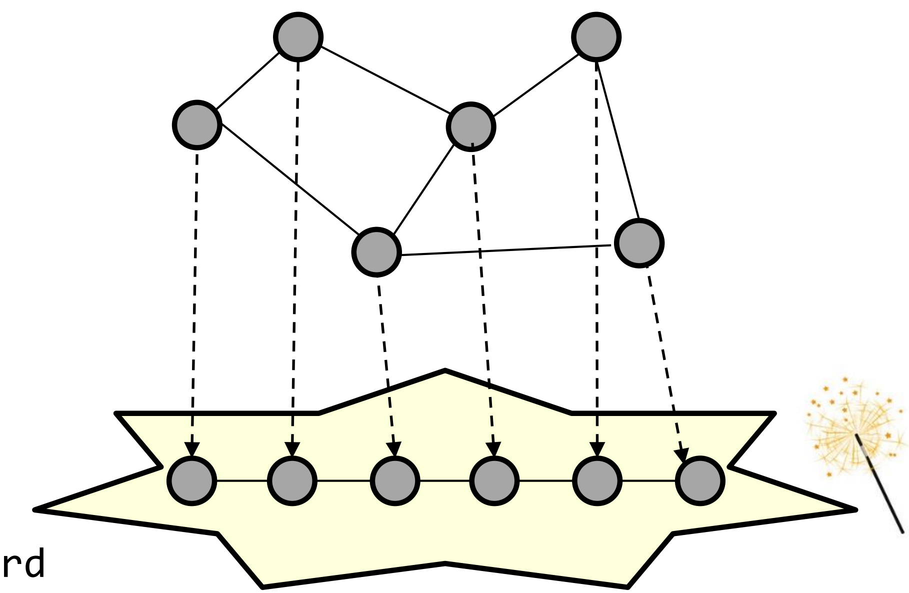
- → Minimizes sum of virtual edges

MLA is **NP-hard**

- → … and so is our problem!

But what about **△>2**?

- → Embedding problem still hard
- → But we have a new **degree of freedom**!



Related Problem

Virtual Network Embedding Problem (VNEP)

Example △=2: A Minium Linear Arrangement (**MLA**) Problem

⇀ Minimizes sum of virtual edges

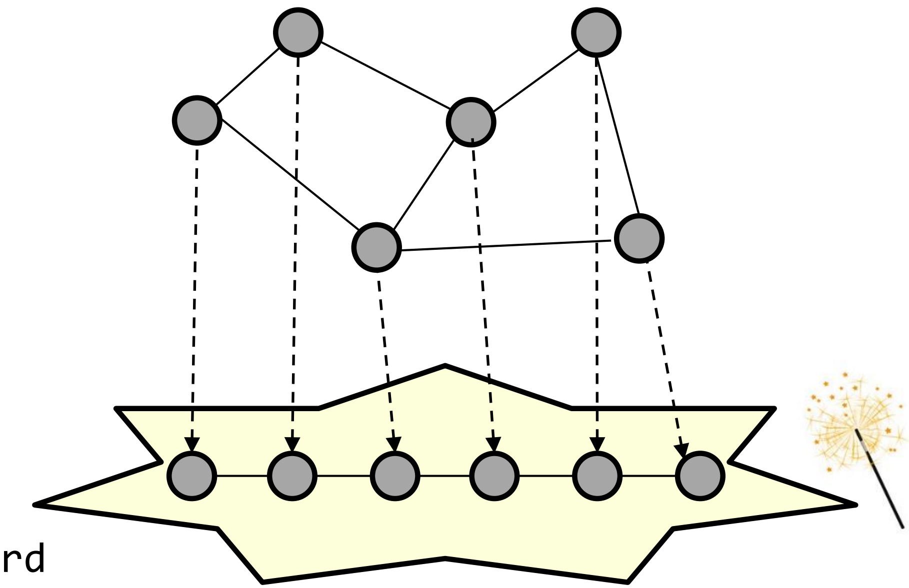
MLA is **NP-hard**

⇀ … and so is our problem!

But what about **△>2**?

⇀ Embedding problem still hard

⇀ But we have a new **degree of freedom**!

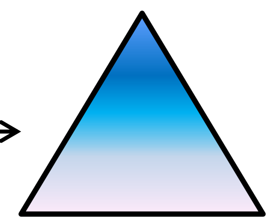


Simplifies problem?!

Entropy Lower Bound

Destinations

Sources	1	2	3	4	5	6	7
1	0	$\frac{2}{65}$	$\frac{1}{13}$	$\frac{1}{65}$	$\frac{1}{65}$	$\frac{2}{65}$	$\frac{3}{65}$
2	$\frac{2}{65}$	0	$\frac{1}{65}$	0	0	0	$\frac{2}{65}$
3	$\frac{1}{13}$	$\frac{1}{65}$	0	$\frac{2}{65}$	0	0	$\frac{1}{13}$
4	$\frac{1}{65}$	0	$\frac{2}{65}$	0	$\frac{4}{65}$	0	0
5	$\frac{1}{65}$	0	$\frac{3}{65}$	$\frac{4}{65}$	0	0	0
6	$\frac{2}{65}$	0	0	0	0	0	$\frac{3}{65}$
	$\frac{3}{65}$	$\frac{2}{65}$	$\frac{1}{13}$	0	0	$\frac{3}{65}$	0



Huffman tree:
"**ego-tree**"

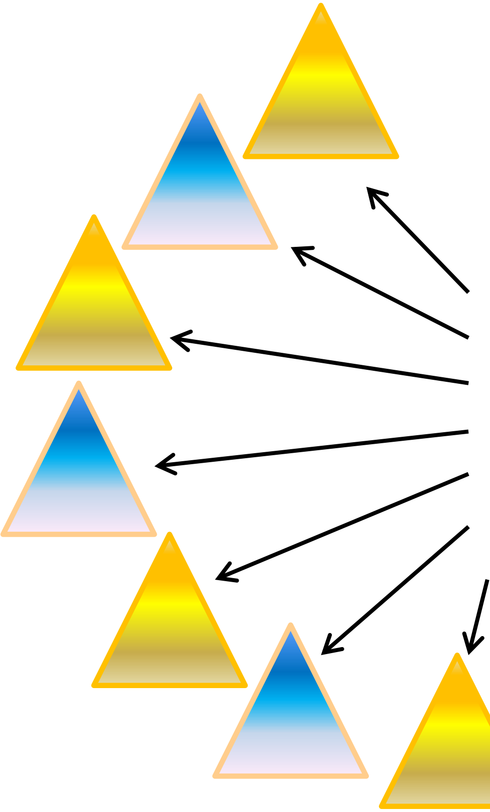
Entropy Lower Bound

destinations

sources

$$\mathrm{ERL} = \Omega(H_\Delta(Y|X))$$

degree

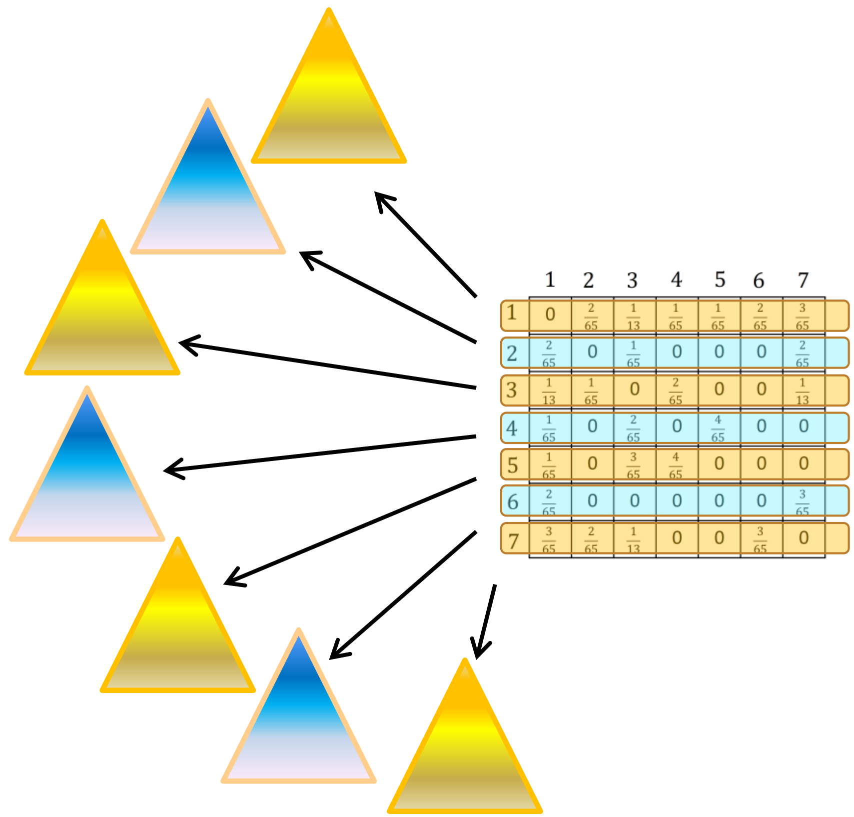


	1	2	3	4	5	6	7
1	0	$\frac{2}{65}$	$\frac{1}{13}$	$\frac{1}{65}$	$\frac{1}{65}$	$\frac{2}{65}$	$\frac{3}{65}$
2	$\frac{2}{65}$	0	$\frac{1}{65}$	0	0	0	$\frac{2}{65}$
3	$\frac{1}{13}$	$\frac{1}{65}$	0	$\frac{2}{65}$	0	0	$\frac{1}{13}$
4	$\frac{1}{65}$	0	$\frac{2}{65}$	0	$\frac{4}{65}$	0	0
5	$\frac{1}{65}$	0	$\frac{3}{65}$	$\frac{4}{65}$	0	0	0
6	$\frac{2}{65}$	0	0	0	0	0	$\frac{3}{65}$
7	$\frac{3}{65}$	$\frac{2}{65}$	$\frac{1}{13}$	0	0	$\frac{3}{65}$	0

Entropy Upper Bound

- Idea for algorithm:
  - union of trees
  - reduce degree
  - but keep distances

Static

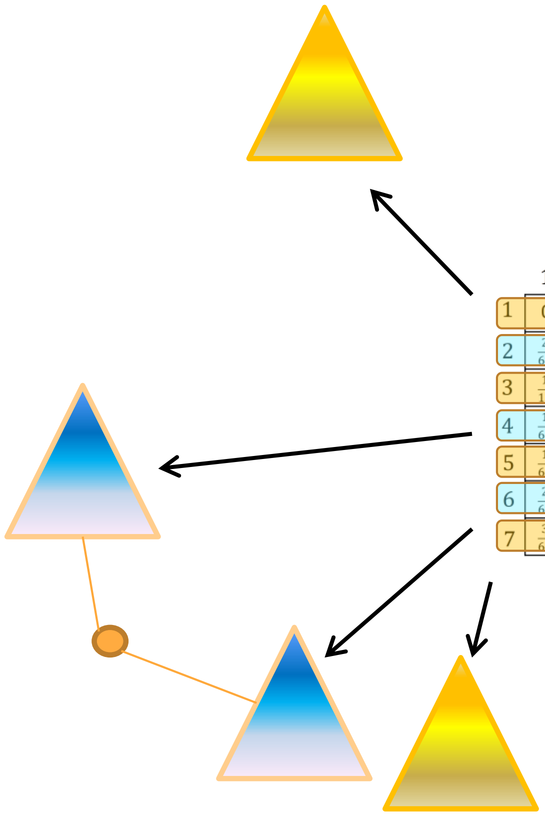


	1	2	3	4	5	6	7
1	0	$\frac{2}{65}$	$\frac{1}{13}$	$\frac{1}{65}$	$\frac{1}{65}$	$\frac{2}{65}$	$\frac{3}{65}$
2	$\frac{2}{65}$	0	$\frac{1}{65}$	0	0	0	$\frac{2}{65}$
3	$\frac{1}{13}$	$\frac{1}{65}$	0	$\frac{2}{65}$	0	0	$\frac{1}{13}$
4	$\frac{1}{65}$	0	$\frac{2}{65}$	0	$\frac{4}{65}$	0	0
5	$\frac{1}{65}$	0	$\frac{3}{65}$	$\frac{4}{65}$	0	0	0
6	$\frac{2}{65}$	0	0	0	0	0	$\frac{3}{65}$
7	$\frac{3}{65}$	$\frac{2}{65}$	$\frac{1}{13}$	0	0	$\frac{3}{65}$	0

Entropy Upper Bound

- Idea for algorithm:
  - union of trees
  - reduce degree
  - but keep distances
- Ok for sparse demands
  - not everyone gets tree
  - helper nodes

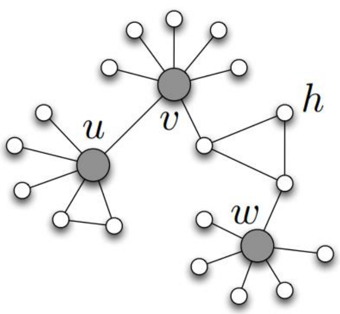
Static



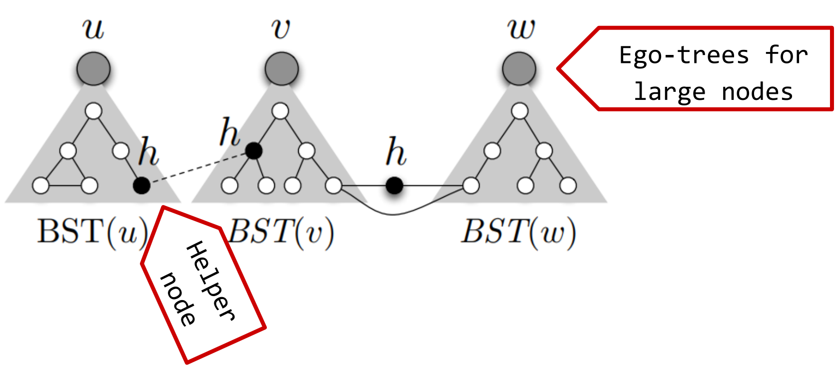
	1	2	3	4	5	6	7
1	0	$\frac{2}{65}$	$\frac{1}{13}$	$\frac{1}{65}$	$\frac{1}{65}$	$\frac{2}{65}$	$\frac{3}{65}$
2	$\frac{2}{65}$	0	$\frac{1}{65}$	0	0	0	$\frac{2}{65}$
3	$\frac{1}{13}$	$\frac{1}{65}$	0	$\frac{2}{65}$	0	0	$\frac{1}{13}$
4	$\frac{1}{65}$	0	$\frac{2}{65}$	0	$\frac{4}{65}$	0	0
5	$\frac{1}{65}$	0	$\frac{3}{65}$	$\frac{4}{65}$	0	0	0
6	$\frac{2}{65}$	0	0	0	0	0	$\frac{3}{65}$
7	$\frac{3}{65}$	$\frac{2}{65}$	$\frac{1}{13}$	0	0	$\frac{3}{65}$	0

Intuition of Algorithm

Demand graph:

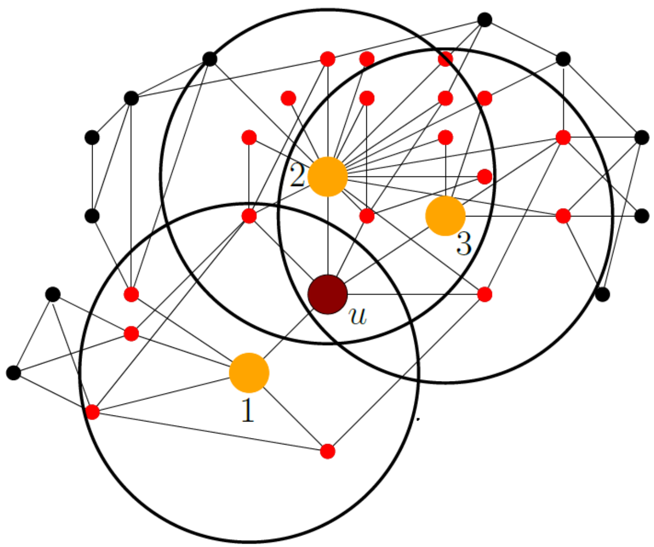


Demand-aware network:



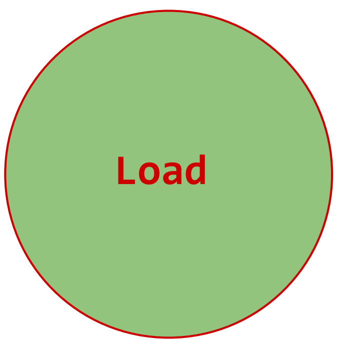
More Optimal Graphs

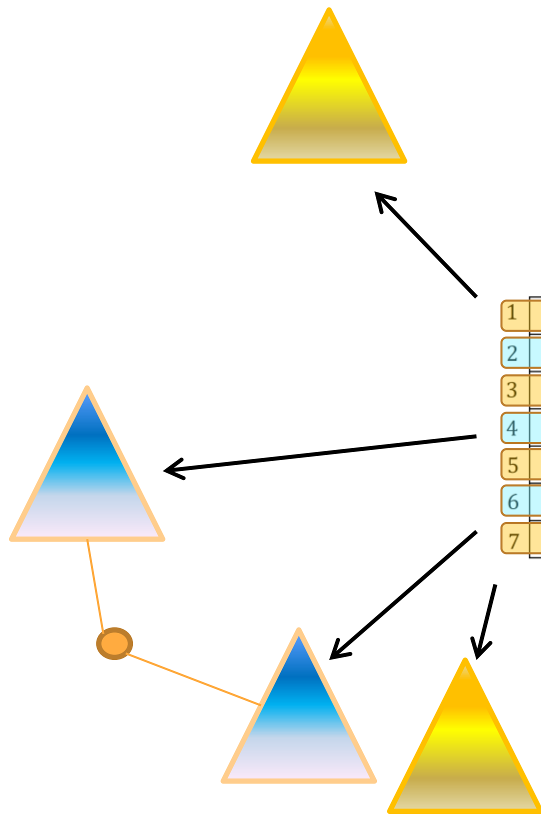
- For regular and uniform demands which admit constant distortion linear spanner
- Graphs of bounded doubling dimension



Accounting for Load

- Still use ego-trees
- But balance for load





	1	2	3	4	5	6	7
1	0	$\frac{2}{65}$	$\frac{1}{13}$	$\frac{1}{65}$	$\frac{1}{65}$	$\frac{2}{65}$	$\frac{3}{65}$
2	$\frac{2}{65}$	0	$\frac{1}{65}$	0	0	0	$\frac{2}{65}$
3	$\frac{1}{13}$	$\frac{1}{65}$	0	$\frac{2}{65}$	0	0	$\frac{1}{13}$
4	$\frac{1}{65}$	0	$\frac{2}{65}$	0	$\frac{4}{65}$	0	0
5	$\frac{1}{65}$	0	$\frac{3}{65}$	$\frac{4}{65}$	0	0	0
6	$\frac{2}{65}$	0	0	0	0	0	$\frac{3}{65}$
7	$\frac{3}{65}$	$\frac{2}{65}$	$\frac{1}{13}$	0	0	$\frac{3}{65}$	0

Further Reading

TON 2016, DISC 2017, CCR 2019, INFOCOM 2019

Demand-Aware Network Designs of Bounded Degree*

Chen Avin[1], Kaushik Mondal[1], and Stefan Schmid[2]

1 Communication Systems Engineering Department
Ben Gurion University of the Negev, Israel
avin@cse.bgu.ac.il, mondal@post.bgu.ac.il
2 Department of Computer Science
Aalborg University, Denmark
schmiste@cs.aau.dk

Abstract

Traditionally, networks such as datacenter interconnects are designed to optimize worst-case performance under *arbitrary* traffic patterns. Such network designs can however be far from optimal when considering the *actual* workloads and traffic patterns which they serve. This insight led to the development of demand-aware datacenter interconnects which can be reconfigured depending on the workload.

Motivated by these trends, this paper initiates the algorithmic study of demand-aware networks (DANs), and in particular the design of bounded-degree networks. The inputs to the network

Toward Demand-Aware Networking: A Theory for Self-Adjusting Networks

Chen Avin
Ben Gurion University, Israel
avin@cse.bgu.ac.il

Stefan Schmid
University of Vienna, Austria
stefan_schmid@univie.ac.at

This article is an editorial note submitted to CCR. It has NOT been peer reviewed.
The authors take full responsibility for this article's technical content. Comments can be posted through CCR Online.

ABSTRACT

The physical topology is emerging as the next frontier in an ongoing effort to render communication networks more flexible. While first empirical results indicate that these flexibilities can be exploited to reconfigure and optimize the network toward the workload it serves and, e.g., providing the same bandwidth at lower infrastructure cost, only little is known today about the fundamental algorithmic problems underlying the design of reconfigurable networks. This paper initiates the study of the theory of demand-aware, self-adjusting networks. Our main position is that self-adjusting networks should be seen through the lense of self-adjusting datas-

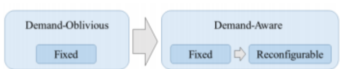


Figure 1: Taxonomy of topology optimization

design of efficient datacenter networks has received much attention over the last years. The topologies underlying mod-

SplayNet: Towards Locally Self-Adjusting Networks

Stefan Schmid*, Chen Avin*, Christian Scheideler, Michael Borokhovich, Bernhard Haeupler, Zvi Lotker

***Abstract*—This paper initiates the study of locally self-adjusting networks: networks whose topology adapts dynamically and in a decentralized manner, to the communication pattern σ. Our vision can be seen as a distributed generalization of the self-adjusting datastructures introduced by Sleator and Tarjan [22]: In contrast to their splay trees which dynamically optimize the lookup costs from a *single node* (namely the tree root), we seek to minimize the routing cost between arbitrary *communication pairs* in the network.**

As a first step, we study distributed binary search trees (BSTs), which are attractive for their support of greedy routing. We introduce a simple model which captures the fundamental tradeoff between the benefits and costs of self-adjusting networks. We present the *SplayNet* algorithm and formally analyze its performance, and prove its optimality in specific case studies. We also introduce lower bound techniques based on interval cuts and

toward static metrics, such as the diameter or the length of the longest route: the self-adjusting paradigm has not spilled over to distributed networks yet.

We, in this paper, initiate the study of a distributed generalization of self-optimizing datastructures. This is a non-trivial generalization of the classic splay tree concept: While in classic BSTs, a *lookup request* always originates from the same node, the tree root, distributed datastructures and networks such as skip graphs [2], [13] have to support *routing requests* between arbitrary pairs (or *peers*) of communicating nodes; in other words, both the source as well as the destination of the requests become variable. Figure 1 illustrates the difference between classic and distributed binary search trees.

In this paper, we ask: Can we reap similar benefits from self-

Demand-Aware Network Design with Minimal Congestion and Route Lengths

Chen Avin
Communication Systems Engineering Dept.
Ben Gurion University of the Negev, Israel

Kaushik Mondal
Communication Systems Engineering Dept.
Ben Gurion University of the Negev, Israel

Stefan Schmid
Faculty of Computer Science
University of Vienna, Austria

***Abstract*—Emerging communication technologies allow to reconfigure the physical network topology at runtime, enabling *demand-aware networks (DANs)*: networks whose topology is optimized toward the workload they serve. However, today, only little is known about the fundamental algorithmic problems underlying the design of such demand-aware networks. This paper presents the first bounded-degree, demand-aware network, *cl-DAN*, which minimizes *both* congestion and route lengths. The designed network is provably (asymptotically) optimal in each dimension individually: we show that there do not exist any bounded-degree networks providing shorter routes (independently of the load), nor do there exist networks providing lower loads (independently of the route lengths). The main building block of the designed *cl-DAN* networks are *ego-trees*: communication sources arrange their communication partners in an optimal tree, *individually*. While the union of these ego-trees forms the basic structure of *cl-DANs*, further techniques are presented to ensure bounded degrees (for scalability).**

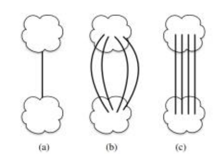


Fig. 1. Challenge of designing demand-aware networks: (a) Optimizing for route lengths only may result in bottlenecks and high loads. (b) Optimizing for congestion only, by distributing load across multiple paths, can result in long routes. (c) Ideally, we aim to design networks that minimize both congestion and route lengths, *using a small number of links* (constant degree).

I. Introduction

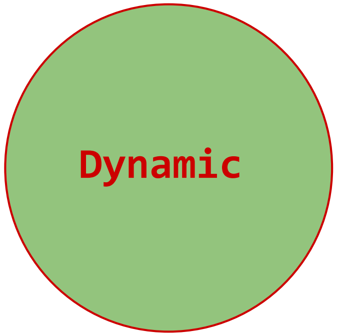
A. Motivation

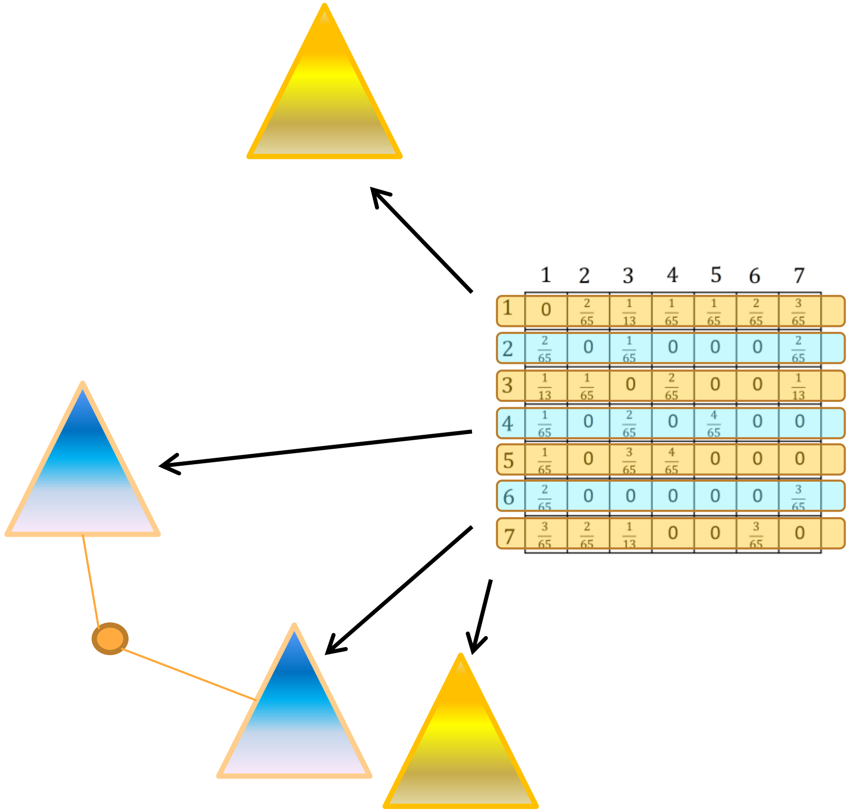
Data center networks have become a critical infrastructure of our digital society. With the trend toward more data-intensive applications, data center network traffic is growing quickly [7], [13]. As much of this traffic is *internal* to the data center (e.g.,

However, only little is known today about the *algorithmic* challenge of designing demand-aware networks which provide low congestion *and* short routes (in the number of hops), for

Dynamic Setting

- Dynamic the same:
  - union of **dynamic ego-trees**
- E.g., SplayNets
- Online algorithms





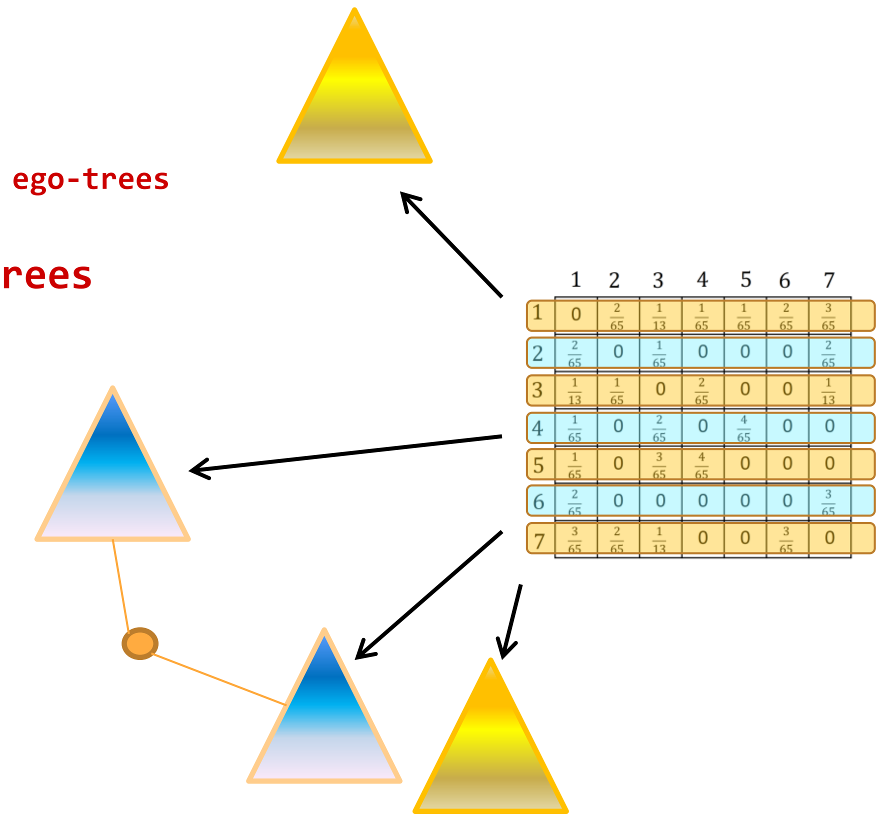
	1	2	3	4	5	6	7
1	0	$\frac{2}{65}$	$\frac{1}{13}$	$\frac{1}{65}$	$\frac{1}{65}$	$\frac{2}{65}$	$\frac{3}{65}$
2	$\frac{2}{65}$	0	$\frac{1}{65}$	0	0	0	$\frac{2}{65}$
3	$\frac{1}{13}$	$\frac{1}{65}$	0	$\frac{2}{65}$	0	0	$\frac{1}{13}$
4	$\frac{1}{65}$	0	$\frac{2}{65}$	0	$\frac{4}{65}$	0	0
5	$\frac{1}{65}$	0	$\frac{3}{65}$	$\frac{4}{65}$	0	0	0
6	$\frac{2}{65}$	0	0	0	0	0	$\frac{3}{65}$
7	$\frac{3}{65}$	$\frac{2}{65}$	$\frac{1}{13}$	0	0	$\frac{3}{65}$	0

Dynamic Setting

$\underbrace{\text{Dynamic}}_{\text{\& distributed}}$

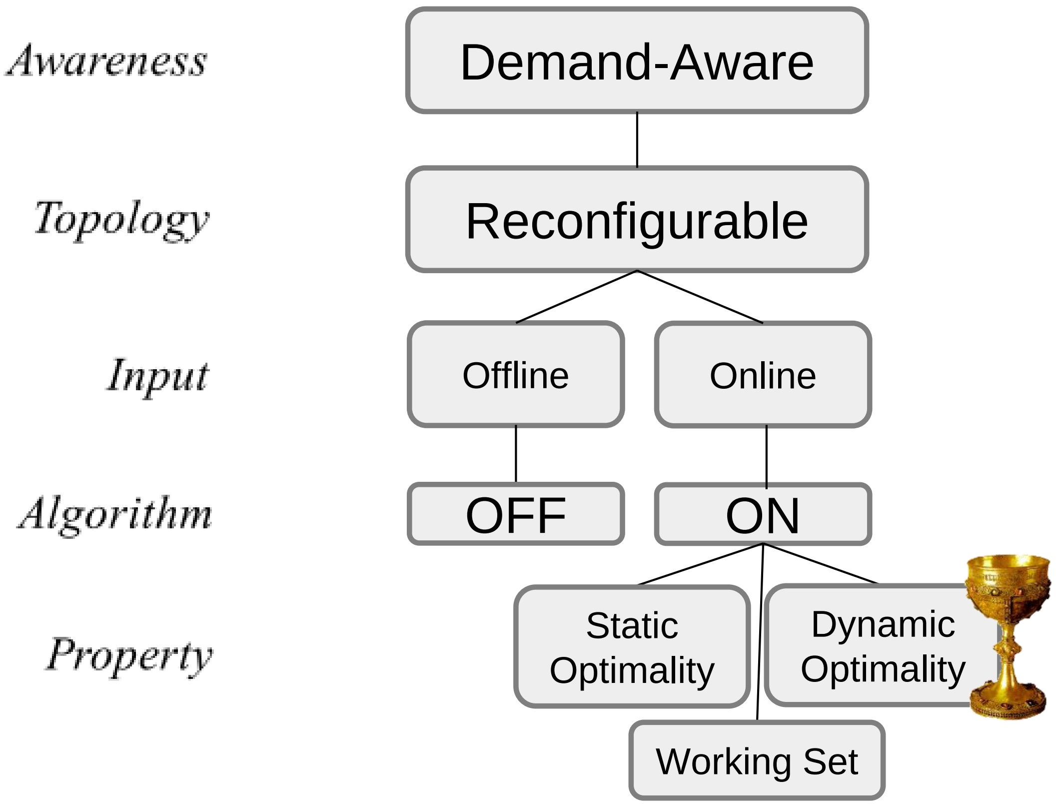
- ⇢ Dynamic the same:
  - ⇀ union of **dynamic&distributed ego-trees**
- ⇢ E.g., SplayNets or **CB trees**
- ⇢ Online algorithms

Dynamic



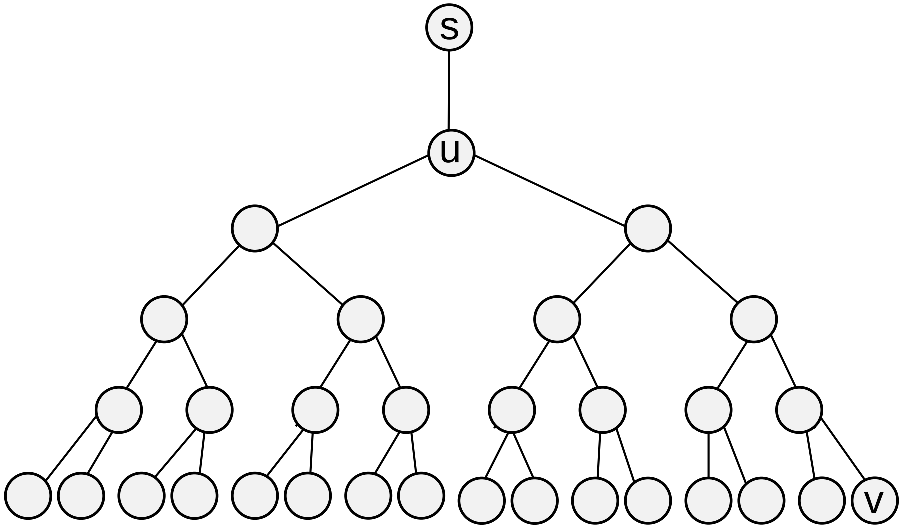
	1	2	3	4	5	6	7
1	0	$\frac{2}{65}$	$\frac{1}{13}$	$\frac{1}{65}$	$\frac{1}{65}$	$\frac{2}{65}$	$\frac{3}{65}$
2	$\frac{2}{65}$	0	$\frac{1}{65}$	0	0	0	$\frac{2}{65}$
3	$\frac{1}{13}$	$\frac{1}{65}$	0	$\frac{2}{65}$	0	0	$\frac{1}{13}$
4	$\frac{1}{65}$	0	$\frac{2}{65}$	0	$\frac{4}{65}$	0	0
5	$\frac{1}{65}$	0	$\frac{3}{65}$	$\frac{4}{65}$	0	0	0
6	$\frac{2}{65}$	0	0	0	0	0	$\frac{3}{65}$
7	$\frac{3}{65}$	$\frac{2}{65}$	$\frac{1}{13}$	0	0	$\frac{3}{65}$	0

Dynamic Objectives



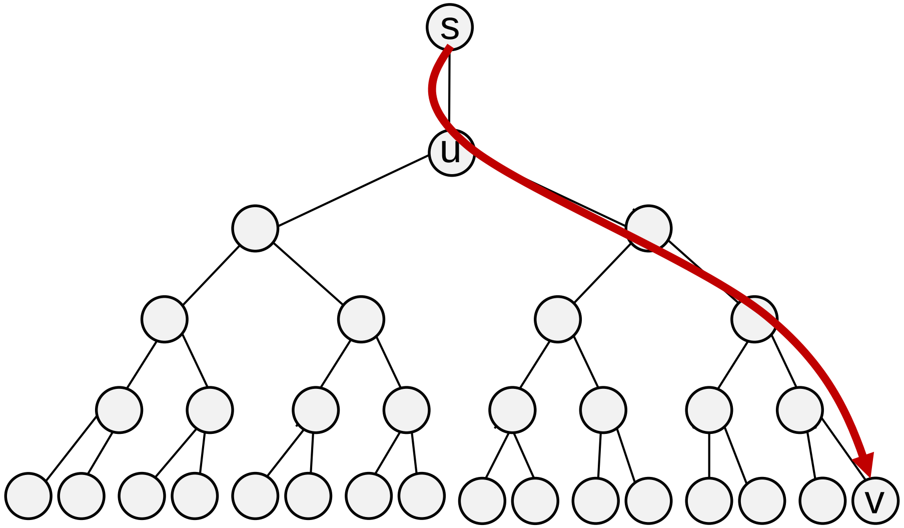
Dynamic Optimality: Push-Down Trees

- For unordered search trees, dynamic optimality is possible: **Push-Down Trees**
- Useful property: **most recently used (MRU)**



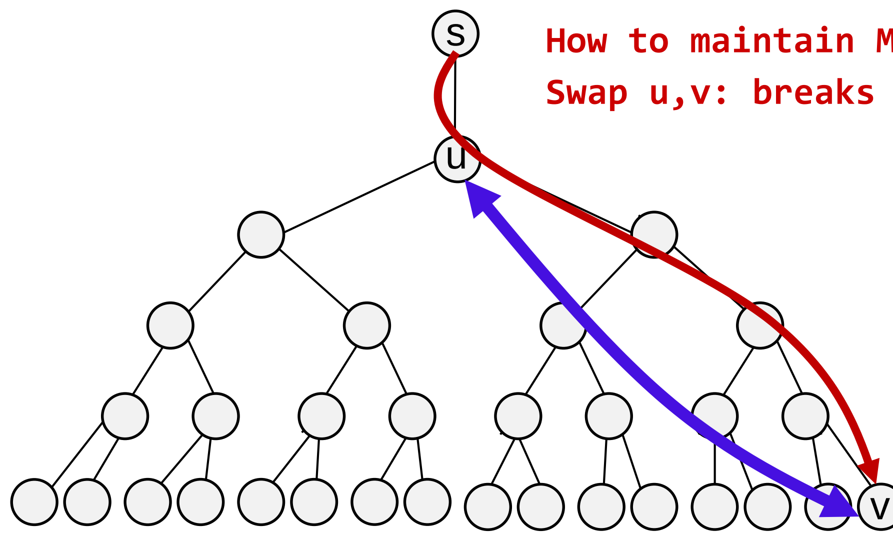
Dynamic Optimality: Push-Down Trees

- For unordered search trees, dynamic optimality is possible: **Push-Down Trees**
- Useful property: **most recently used (MRU)**



Dynamic Optimality: Push-Down Trees

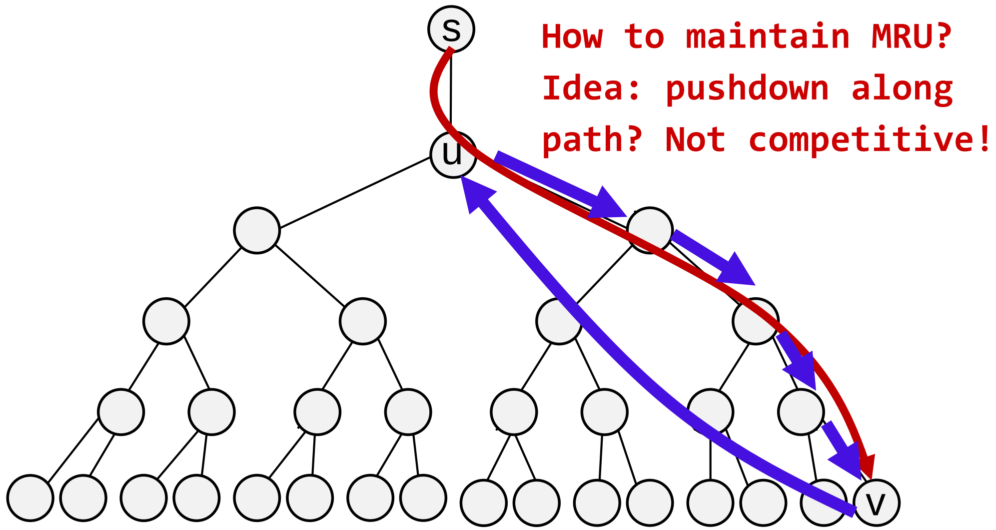
- For unordered search trees, dynamic optimality is possible: **Push-Down Trees**
- Useful property: **most recently used (MRU)**



How to maintain MRU?
Swap u,v: breaks MRU!

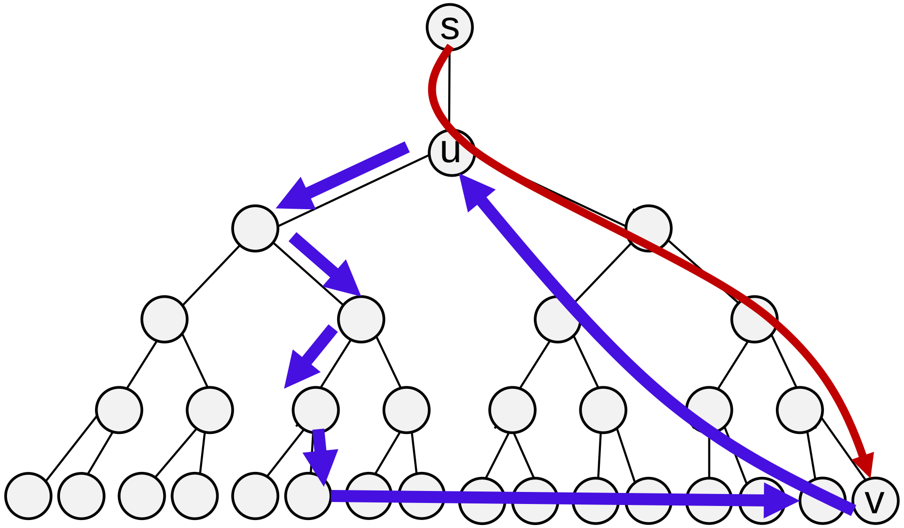
Dynamic Optimality: Push-Down Trees

- For unordered search trees, dynamic optimality is possible: **Push-Down Trees**
- Useful property: **most recently used (MRU)**



Dynamic Optimality: Push-Down Trees

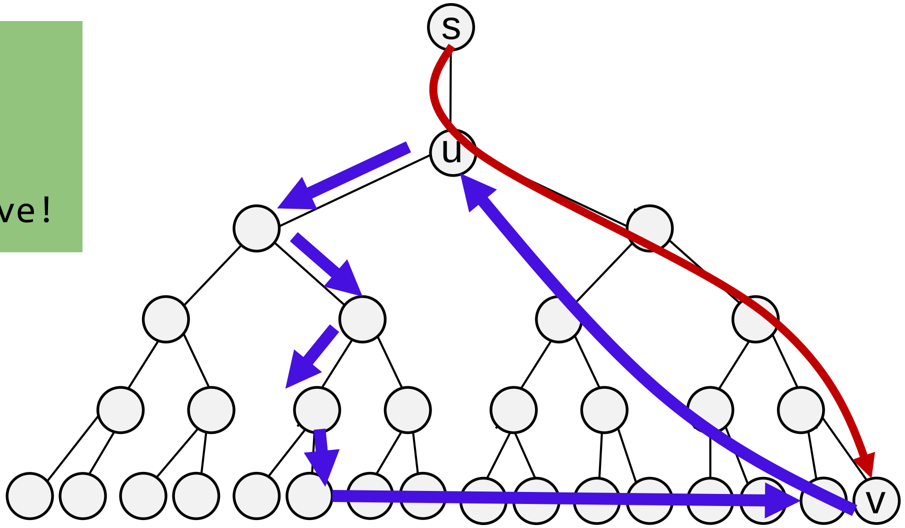
- For unordered search trees, dynamic optimality is possible: **Push-Down Trees**
- Useful property: **most recently used (MRU)**
- Idea: **balanced** pushdown (random vs deterministic?)



Dynamic Optimality: Push-Down Trees

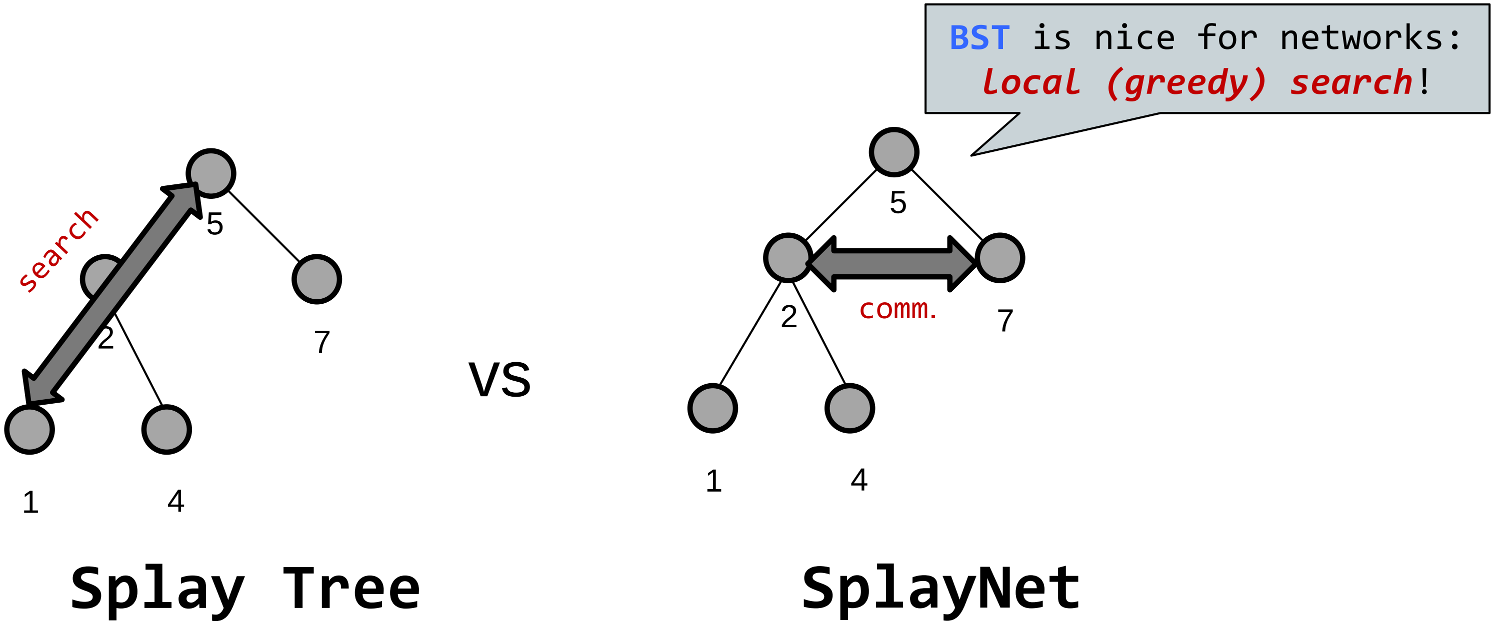
- For unordered search trees, dynamic optimality is possible: **Push-Down Trees**
- Useful property: **most recently used (MRU)**
- Idea: **balanced** pushdown (random vs deterministic?)

Random walk preservers MRU: constant competitive. Deterministic does not, but still constant competitive!

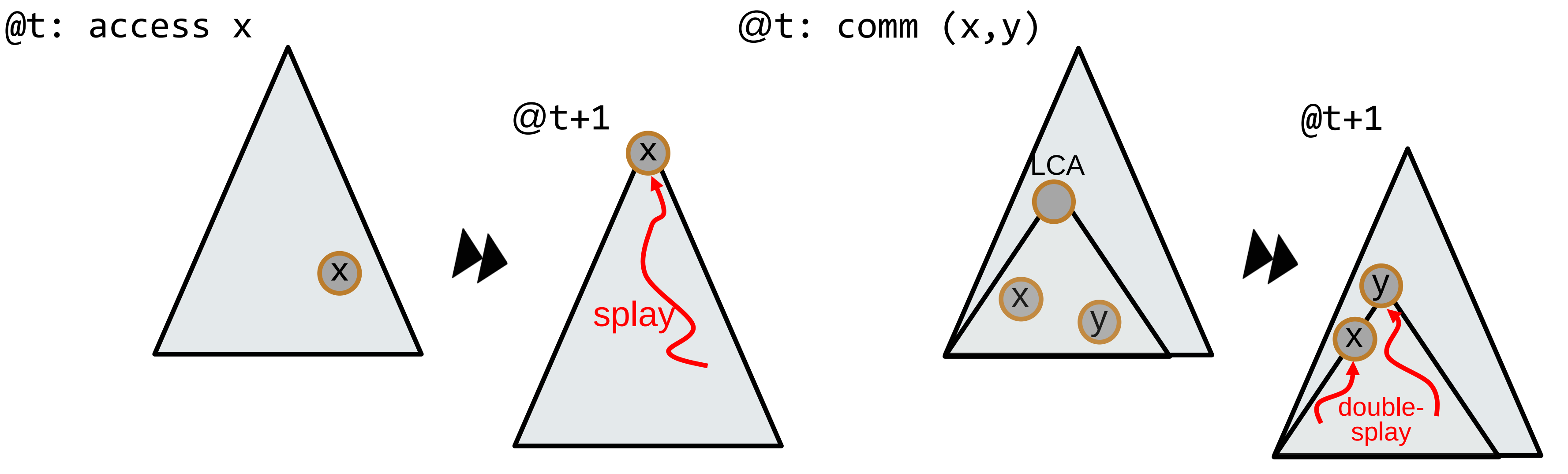


An Alternative: SplayNets

⇢ Idea: generalize splay trees to networks



SplayNets: A Simple Idea



Splay Tree

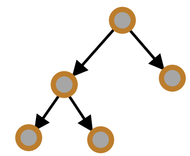
SplayNet

Properties of SplayNets

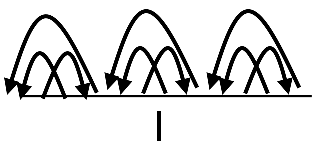
- Statically optimal if demand comes from a ***product distribution***
  - Product distribution: entropy equals conditional entropy, i.e., H(X)+H(Y)=H(X|Y)+H(X|Y)

- Converges to optimal static topology in
  - **Multicast scenario**: requests come from a binary tree as well
  - **Cluster scenario**: communication only within interval
  - **Laminated scenario** : communication is „non-crossing matching"

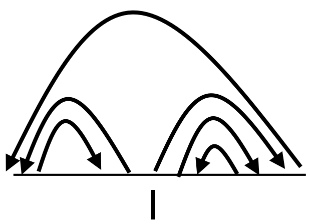
Multicast Scenario



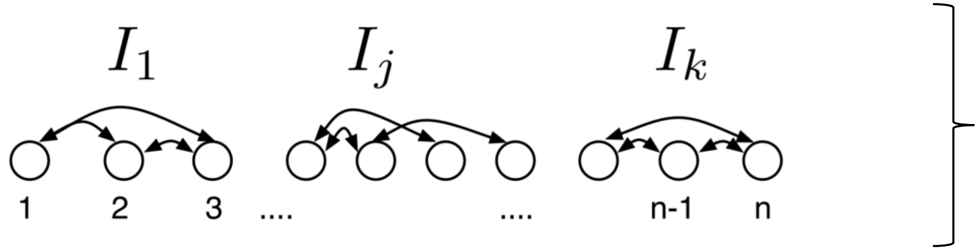
Cluster Scenario



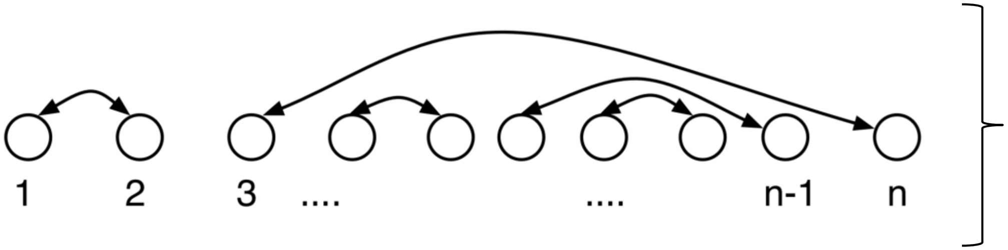
Laminated Scenario



More Specifically



Cluster scenario: SplayNet will converge to state where ***paths between cluster nodes only includes cluster nodes***



Non-crossing matching scenario: SplayNet will converge to state where ***all communication pairs are adjacent***

Further Reading

TON 2016, LATIN 2020, IPDPS 2021

SplayNet: Towards Locally Self-Adjusting Networks

Stefan Schmid*, Chen Avin*, Christian Scheideler, Michael Borokhovich, Bernhard Haeupler, Zvi Lotker

***Abstract*—This paper initiates the study of locally self-adjusting networks: networks whose topology adapts dynamically and in a decentralized manner, to the communication pattern σ. Our vision can be seen as a distributed generalization of the self-adjusting datastructures introduced by Sleator and Tarjan [22]: In contrast to their splay trees which dynamically optimize the lookup costs from a *single node* (namely the tree root), we seek to minimize the routing cost between arbitrary *communication pairs* in the network.**

As a first step, we study distributed binary search trees (BSTs), which are attractive for their support of greedy routing

toward static metrics, such as the diameter or the length of the longest route: the self-adjusting paradigm has not spilled over to distributed networks yet.

We, in this paper, initiate the study of a distributed generalization of self-optimizing datastructures. This is a non-trivial generalization of the classic splay tree concept: While in classic BSTs, a *lookup request* always originates from the same node, the tree root, distributed datastructures and networks such as skip graphs [2], [13] have to support *routing requests*

Dynamically Optimal Self-Adjusting Single-Source Tree Networks

Chen Avin[1], Kaushik Mondal[2], and Stefan Schmid[3]

[1] Ben Gurion University of the Negev, Israel
[2] Indian Institute of Technology Ropar, India
[3] Faculty of Computer Science, University of Vienna, Austria

Abstract. This paper studies a fundamental algorithmic problem related to the design of demand-aware networks: networks whose topologies adjust toward the traffic patterns they serve, in an online manner. The goal is to strike a tradeoff between the benefits of such adjustments (shorter routes) and their costs (reconfigurations). In particular, we consider the problem of designing a self-adjusting tree network which serves single-source, multi-destination communication. The problem has

CBNet: Minimizing Adjustments in Concurrent Demand-Aware Tree Networks

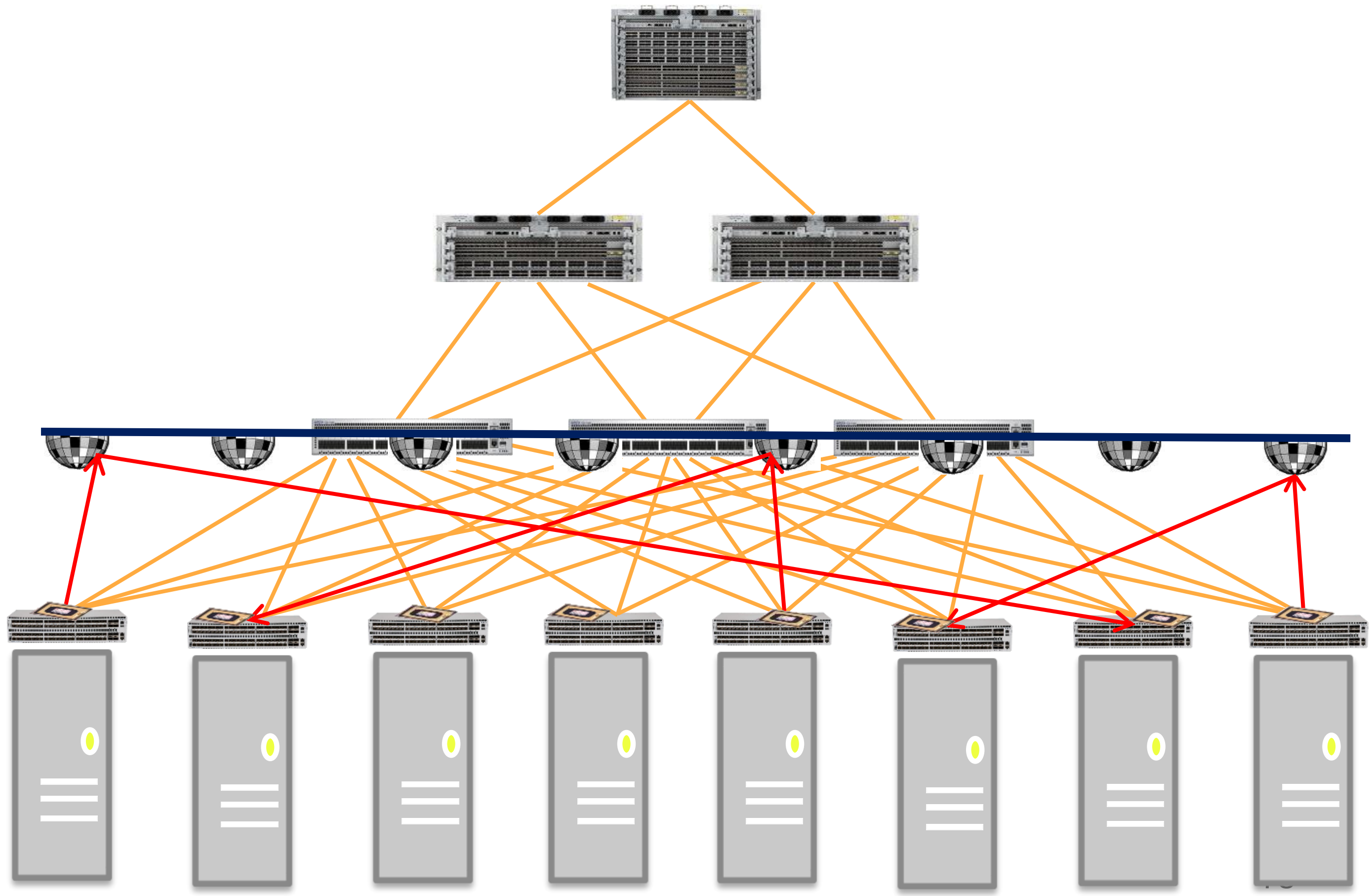
Otavio Augusto de Oliveira Souza[1] Olga Goussevskaia[1] Stefan Schmid[2]
[1] Universidade Federal de Minas Gerais, Brazil [2] University of Vienna, Austria

***Abstract*—This paper studies the design of demand-aware network topologies: networks that dynamically adapt themselves toward the demand they currently serve, in an online manner. While demand-aware networks may be significantly more efficient than demand-oblivious networks, frequent adjustments are still costly. Furthermore, a centralized controller of such networks may become a bottleneck.**

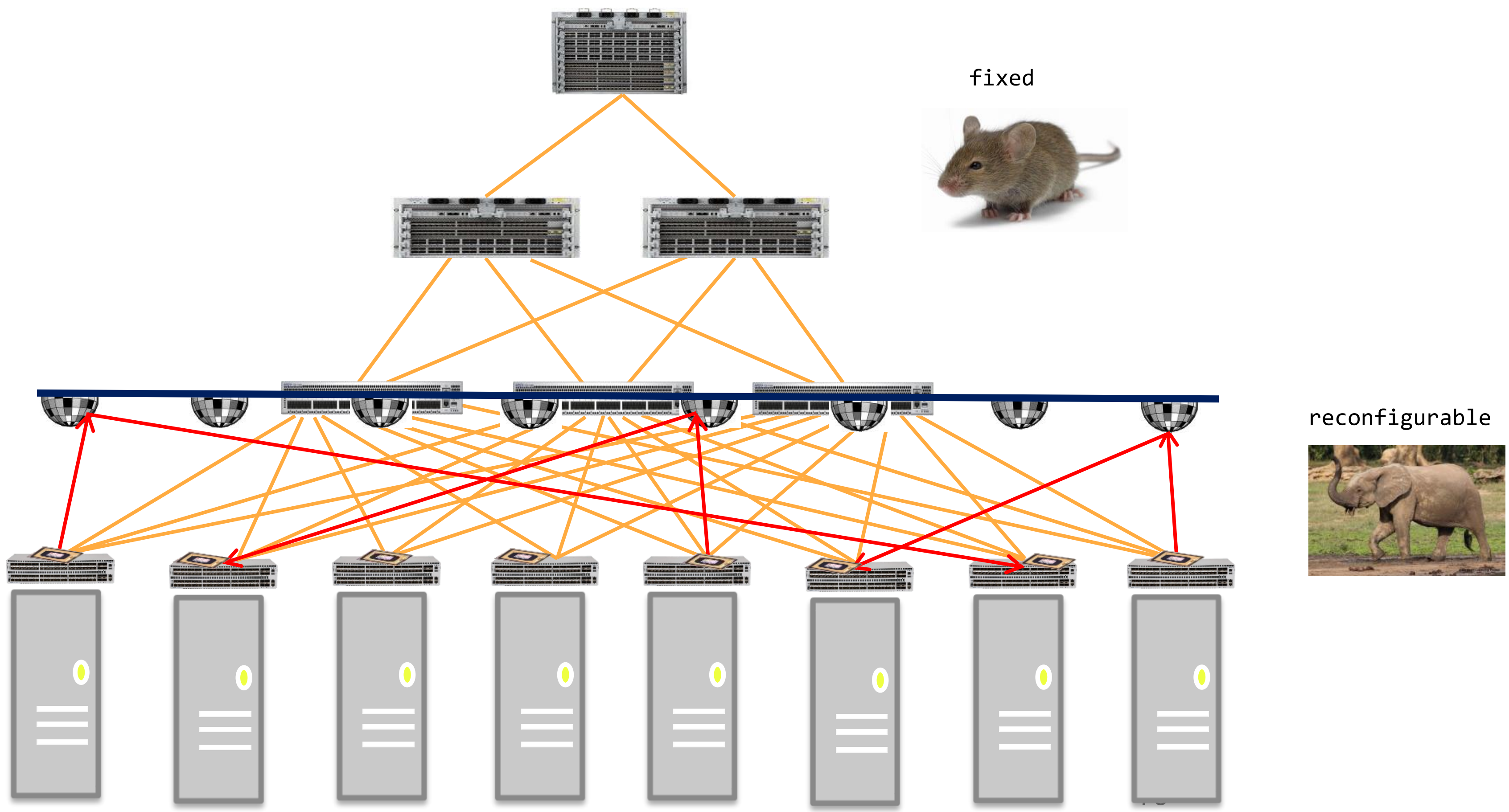
We present CBNet (Counting-Based self-adjusting Network), a

CBNet is based on concepts from self-adjusting data structures, and in particular, CBTrees [12]. CBNet gradually adapts the network topology toward the communication pattern in an online manner, i.e., without previous knowledge of the demand distribution. At the same time, *bidirectional semi-splaying* and counters are used to maintain state, minimize reconfiguration costs and maximize concurrency.

Hybrid Networks



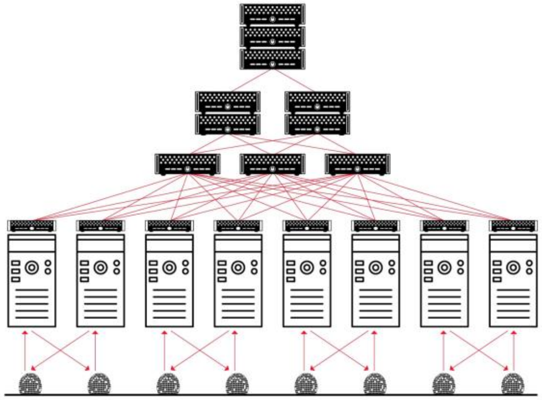
Hybrid Networks



ReNet

A Statically Optimal Demand-Aware Network

- Model: **hybrid architecture**
  - Fixed network of diameter log n plus reconfigurable network (**constant** number of direct links)
  - **Segregated** routing
  - **Online** sequence of requests: σ = (σ1, σ2, σ3, ...)
  - Global controller



fixed

reconfigurable

- **Objective:** Minimize route length plus reconfigurations
  - More specifically: be **statically optimal**
  - Compared to a fixed algorithm which knows σ ahead of time

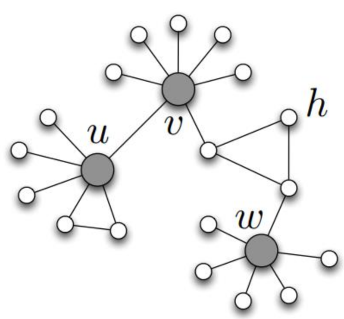
BONUS

- Compact routing (constant tables)
- Local routing (greedy)
- Arbitrary addressing

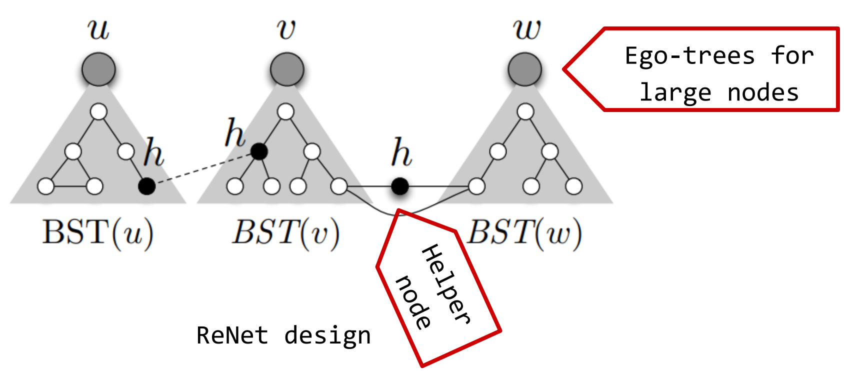
The ReNet Algorithm (1)

Algorithmic building blocks:

1. **Working Set** (WS)
   ⇀ Nodes keep track of recent communication partners in σ.
2. Small/large nodes and **Ego-Tree**
   ⇀ Nodes with small WS connect to WS directly, nodes with large WS via a self-adjusting binary search tree (e.g., a **splay tree**)
3. **Helper nodes** to reduce the degree
   ⇀ Large nodes may appear in many ego-trees, so get help of small nodes



Demand graph

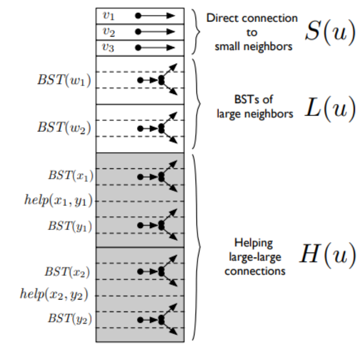


ReNet design

The ReNet Algorithm (2)

Continued:

4. **Self adjustments**
   - Keep track of WS; when too large: **flush-when-full**
5. Centralized coordination
   - Fairly **decentralized**: coordinator only needs to keep track of which nodes are large and which small
   - Nodes inform coordinator when adding node to working set
   - Coordinator then assigns helper node on demand



Analytical Results (1)

Theorem 1:

For any sparse communication sequence of a certain length, ReNets are statically optimal while ensuring a bounded degree.

⇢ Sparse: subsequences of only involve a linear number of nodes

⇢ Required to ensure availability of helper nodes (DISC 2017)

Analytical Results (2)

Theorem 2:

Under certain communication patterns, the amortized cost of ReNet can be significantly lower than the static optimum, i.e., Ω(log n).

- Example: consider sequence of $\sigma = (\sigma^{(1)}, \sigma^{(2)}, \sigma^{(3)}, ...)$ where each $\sigma^{(i)}$ is of length n log n, sparse and corresponds to different **2-dimensional grid**.
- In this example, the cost of ReNet is **constant** for each $\sigma^{(i)}$.
- Overall, the union of the grids form a uniform pattern, so the cost of the static algorithm is **log n** (for constant degree).

Further Reading

PERFORMANCE 2020, SPAA 2021, APOCS 2021

Online Dynamic B-Matching

With Applications to Reconfigurable Datacenter Networks

Marcin Bienkowski
University of Wrocław, Poland
marcin.bienkowski@cs.uni.wroc.pl

David Fuchssteiner
University of Vienna, Austria
david.alexander.fuchssteiner@univie.ac.at

Jan Marcinkowski
University of Wrocław, Poland
jan.marcinkowski@cs.uni.wroc.pl

Stefan Schmid
University of Vienna, Austria
stefan_schmid@univie.ac.at

ABSTRACT

This paper initiates the study of online algorithms for the maximum weight b-matching problem, a generalization of maximum weight matching where each node has at most $b \geq 1$ adjacent matching edges. The problem is motivated by emerging optical technologies which allow to enhance datacenter networks with reconfigurable matchings, providing direct connectivity between frequently communicating racks. These additional links may improve network per-

An emerging intriguing alternative to these static datacenter networks are reconfigurable networks [11, 13, 26, 31, 32, 40, 43, 50, 51, 64, 65, 68]: networks whose topology can be changed *dynamically*. In particular, novel optical technologies allow to provide "short cuts", i.e., direct connectivity between top-of-rack switches, based on dynamic matchings. First empirical studies demonstrate the potential of such reconfigurable networks, which can deliver very high bandwidth efficiency at low cost.

ReNets: Statically-Optimal Demand-Aware Networks*

Chen Avin† Stefan Schmid‡

Abstract

This paper studies the design of *self-adjusting* datacenter networks whose physical topology dynamically adapts to the workload, in an *online* and *demand-aware* manner. We propose *ReNet*, a self-adjusting network which does not require any predictions about future demands and amortizes reconfigurations: it performs as good as a hypothetical static algorithm with perfect knowledge of the future demand. In particular, we show that for arbitrary *sparse* communication demands, *ReNets* achieve *static optimality*, a fundamental property of learning algorithms, and that route lengths in *ReNets* are proportional to existing lower bounds, which are known to relate to an *entropy* metric of the demand. *ReNets* provide additional desirable properties such as *compact* and *local* routing and flat addressing therefore ensuring scalability and further reducing the overhead of reconfiguration. To achieve these properties, *ReNets* combine

we consider the design of DANs which provide short average route lengths by accounting for locality in the demand and by locating frequently communicating node pairs (e.g., a pair of top-of-the-rack switches) topologically closer. Shorter routes can improve network performance (e.g., latency) and reduce costs (e.g., load, energy consumption) [6].

DANs come in two flavors: *fixed* and *self-adjusting*. Fixed DANs can exploit *spatial* locality in the demand. It has recently been shown that a fixed DAN can provide average route lengths in the order of the (conditional) *entropy* of the demand [7,8,9], which can be, *for specific demands*, much lower than the $O(\log n)$ route lengths provided by demand-*oblivious* networks. However, fixed DANs require *a priori* knowledge of the demand.

On the contrary, *self-adjusting* DANs do not require such knowledge and can additionally exploit *temporal*

Scheduling Opportunistic Links in Two-Tiered Reconfigurable Datacenters

Janardhan Kulkarni
Microsoft Research, Redmond, USA
jakul@microsoft.com

Stefan Schmid
University of Vienna, Austria
stefan_schmid@univie.ac.at

Paweł Schmidt
University of Wrocław, Poland
pawel.schmidt@cs.uni.wroc.pl

***Abstract*—Reconfigurable optical topologies are emerging as a promising technology to improve the efficiency of datacenter networks. This paper considers the problem of scheduling opportunistic links in such reconfigurable datacenters. We study the online setting and aim to minimize flow completion times. The problem is a two-tier generalization of classic switch scheduling problems. We present a stable-matching algorithm which is $2 \cdot (2/\varepsilon + 1)$-competitive against an optimal offline algorithm, in a resource augmentation model: the online algorithm runs**

particular, we consider a two-stage switch scheduling model as it arises in existing datacenter architectures, e.g., based on free-space optics [11]. In a nutshell (a formal model will follow shortly), we consider an architecture where traffic demands (modelled as *packets*) arise between Top-of-Rack (ToR) switches, while opportunistic links are between lasers and photodetectors, and where many laser-photodetector combinations can serve traffic between a pair of ToRs. The goal is

Future Work: Models, Metrics, Algos

so far

scratched surface

to do ☺

Notion of self-adjusting networks opens a **large uncharted field** with many questions:

→ Metrics and algorithms: by how much can load be lowered, **energy** reduced, quality-of-service improved, etc. in demand-aware networks? Even for **route length** not clear!

→ How to **model** reconfiguration costs?

→ Impact on **other layers**?

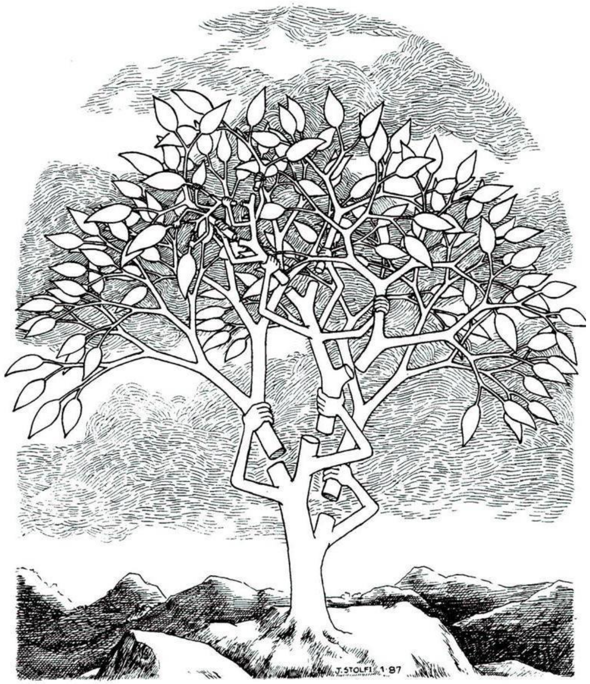
Requires knowledge in networking, distributed systems, algorithms, performance evaluation.

Conclusion

- Demand-aware networks
  - Much potential…
  - … if demand has structure
  - Metrics? E.g., entropy

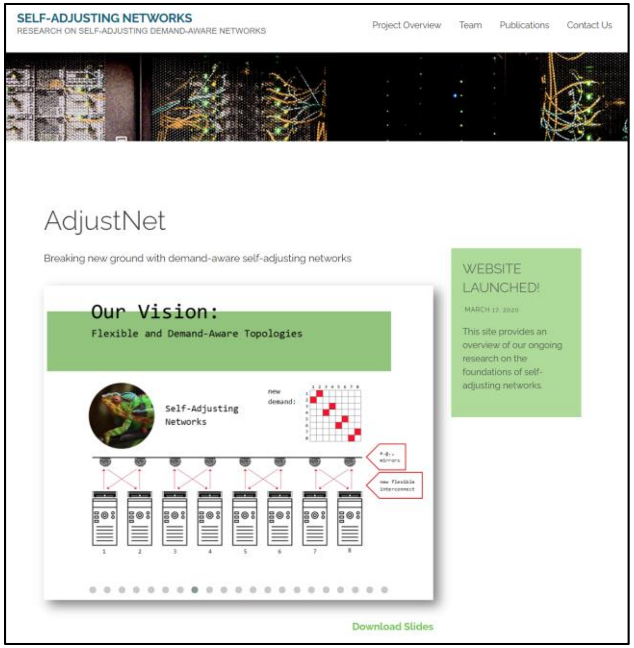
- Avenues for future work
  - Dense communication
  - Dynamic optimality
  - Distributed control plane

Thank you!

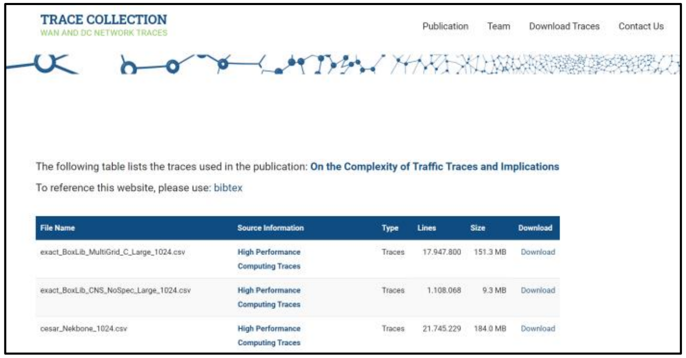


A Self-Adjusting Search Tree
by Jorge Stolfi (1987)

Websites



http://self-adjusting.net/
Project website



https://trace-collection.net/
Trace collection website

Further Reading

Static DAN

Demand-Aware Network Designs of Bounded Degree

Chen Avin Kaushik Mondal Stefan Schmid

Abstract Traditionally, networks such as datacenter interconnects are designed to optimize worst-case performance under *arbitrary* traffic patterns. Such network designs can however be far from optimal when considering the *actual* workloads and traffic patterns which they serve. This insight led to the development of demand-aware datacenter interconnects which can be reconfigured depending on the workload.

Motivated by these trends, this paper initiates the algorithmic study of demand-aware networks (DANs), and in particular the design of bounded-degree networks. The inputs to the network design problem are a discrete communication request distribution, $\mathcal{D}$, defined over communicating pairs from the node set V, and a bound, Δ, on the maximum degree. In turn, our objective is to design an (undirected) demand-aware network $N = (V, E)$ of bounded-degree Δ, which provides short routing paths between frequently communicating nodes distributed across N. In particular, the designed network should minimize the *expected path length* on N (with respect to $\mathcal{D}$), which is a basic measure of the

1 Introduction

The problem studied in this paper is motivated by the advent of more flexible datacenter interconnects, such as ProjecToR [29,31]. These interconnects aim to overcome a fundamental drawback of traditional datacenter network designs: the fact that network designers must decide *in advance* on how much capacity to provision between electrical packet switches, e.g., between Top-of-Rack (ToR) switches in datacenters. This leads to an undesirable tradeoff [42]: either capacity is over-provisioned and therefore the interconnect expensive (e.g., a fat-tree provides full-bisection bandwidth), or one may risk congestion, resulting in a poor cloud application performance. Accordingly, systems such as ProjecToR provide a reconfigurable interconnect, allowing to establish links flexibly and in a *demand-aware manner*. For example, direct links or at least short communication paths can be established between frequently communicating ToR switches. Such links can be implemented using a bounded number of lasers, mirrors,

Overview: Models

Toward Demand-Aware Networking: A Theory for Self-Adjusting Networks

Chen Avin
Ben Gurion University, Israel
avin@cse.bgu.ac.il

Stefan Schmid
University of Vienna, Austria
stefan_schmid@univie.ac.at

This article is an editorial note submitted to CCR. It has NOT been peer reviewed.
The authors take full responsibility for this article's technical content. Comments can be posted through CCR Online.

ABSTRACT

The physical topology is emerging as the next frontier in an ongoing effort to render communication networks more flexible. While first empirical results indicate that these flexibilities can be exploited to reconfigure and optimize the network toward the workload it serves and, e.g., providing the same bandwidth at lower infrastructure cost, only little is known today about the fundamental algorithmic problems underlying the design of reconfigurable networks. This paper initiates the study of the theory of demand-aware, self-adjusting networks. Our main position is that self-adjusting networks should be seen through the lense of self-adjusting datastructures. Accordingly, we present a taxonomy classifying the different algorithmic models of demand-oblivious, fixed demand-aware, and reconfigurable demand-aware networks, introduce a formal model, and identify objectives and evaluation metrics. We also demonstrate, by examples, the inherent

Figure 1: Taxonomy of topology optimization

design of efficient datacenter networks has received much attention over the last years. The topologies underlying modern datacenter networks range from trees [7, 8] over hypercubes [9, 10] to expander networks [11] and provide high connectivity at low cost [1].

Until now, these networks also have in common that their topology is *fixed* and ***oblivious*** to the actual demand (i.e.,

Static Optimality

ReNets: Toward Statically Optimal Self-Adjusting Networks

Chen Avin[1] Stefan Schmid[2]
[1] Ben Gurion University, Israel [2] University of Vienna, Austria

Abstract

This paper studies the design of *self-adjusting* networks whose topology dynamically adapts to the workload, in an *online* and *demand-aware* manner. This problem is motivated by emerging optical technologies which allow to reconfigure the datacenter topology at runtime. Our main contribution is *ReNet*, a self-adjusting network which maintains a balance between the benefits and costs of reconfigurations. In particular, we show that *ReNets* are *statically optimal* for arbitrary sparse communication demands, i.e., perform at least as good as any fixed demand-aware network designed with a perfect knowledge of the *future* demand. Furthermore, *ReNets* provide *compact* and *local* routing, by leveraging ideas from self-adjusting datastructures.

1 Introduction

Modern datacenter networks rely on efficient network topologies (based on fat-trees [1], hypercubes [2, 3], or expander [4] graphs) to provide a high connectivity at low cost [5]. These datacenter networks have in common that their topology is *fixed* and *oblivious* to the actual demand (i.e., workload or communication pattern) they currently serve. Rather, they are designed for all-to-all communication patterns, by ensuring properties such as full bisection bandwidth or $O(\log n)$ route lengths between *any* node pair in a constant-degree n-node network. However, demand-oblivious networks can be inefficient for more *specific* demand patterns, as they usually arise in practice. Empirical studies show that traffic patterns in datacenters are often

Robust DAN

***rDAN*: Toward Robust Demand-Aware Network Designs**

Chen Avin[1] Alexandr Hercules[1] Andreas Loukas[2] Stefan Schmid[3]
[1] Ben-Gurion University, IL [2] EPFL, CH [3] University of Vienna, AT & TU Berlin, DE

Abstract

We currently witness the emergence of interesting new network topologies optimized towards the traffic matrices they serve, such as demand-aware datacenter interconnects (e.g., ProjecToR) and demand-aware peer-to-peer overlay networks (e.g., SplayNets). This paper introduces a formal framework and approach to reason about and design robust demand-aware networks (*DAN*). In particular, we establish a connection between the communication frequency of two nodes and the path length between them in the network, and show that this relationship depends on the *entropy* of the communication matrix. Our main contribution is a novel robust, yet sparse, family of networks, short *rDAN*s, which guarantee an expected path length that is proportional to the entropy of the communication patterns.

Dynamic DAN

SplayNet: Towards Locally Self-Adjusting Networks

Stefan Schmid*, Chen Avin*, Christian Scheideler, Michael Borokhovich, Bernhard Haeupler, Zvi Lotker

***Abstract*—This paper initiates the study of locally self-adjusting networks: networks whose topology adapts dynamically and in a decentralized manner, to the communication pattern σ. Our vision can be seen as a distributed generalization of the self-adjusting datastructures introduced by Sleator and Tarjan [22]: In contrast to their splay trees which dynamically optimize the lookup costs from a *single node* (namely the tree root), we seek to minimize the routing cost between arbitrary *communication pairs* in the network.**

As a first step, we study distributed binary search trees (BSTs), which are attractive for their support of greedy routing. We introduce a simple model which captures the fundamental tradeoff between the benefits and costs of self-adjusting networks. We present the *SplayNet* algorithm and formally analyze its performance, and prove its optimality in specific case studies. We also introduce lower bound techniques based on interval cuts and edge expansion, to study the limitations of any demand-optimized network. Finally, we extend our study to multi-tree networks, and highlight an intriguing difference between classic and distributed splay trees.

I. Introduction

In the 1980s, Sleator and Tarjan [22] proposed an appealing new paradigm to design efficient Binary Search Tree (BST) datastructures: rather than optimizing traditional metrics such toward static metrics, such as the diameter or the length of the longest route: the self-adjusting paradigm has not spilled over to distributed networks yet.

We, in this paper, initiate the study of a distributed generalization of self-optimizing datastructures. This is a non-trivial generalization of the classic splay tree concept: While in classic BSTs, a *lookup request* always originates from the same node, the tree root, distributed datastructures and networks such as skip graphs [2], [13] have to support *routing requests* between arbitrary pairs (or *peers*) of communicating nodes; in other words, both the source as well as the destination of the requests become variable. Figure 1 illustrates the difference between classic and distributed binary search trees.

In this paper, we ask: Can we reap similar benefits from self-adjusting *entire networks*, by adaptively reducing the distance between frequently communicating nodes?

As a first step, we explore fully decentralized and self-adjusting Binary Search Tree networks: in these networks, nodes are arranged in a binary tree which respects node identifiers. A BST topology is attractive as it supports greedy routing: a node can decide locally to which port to forward a request given its destination address.

Concurrent DANs

CBNet: Minimizing Adjustments in Concurrent Demand-Aware Tree Networks

Otavio Augusto de Oliveira Souza[1] Olga Goussevskaia[1] Stefan Schmid[2]
[1] Universidade Federal de Minas Gerais, Brazil [2] University of Vienna, Austria

***Abstract*—This paper studies the design of demand-aware network topologies: networks that dynamically adapt themselves toward the demand they currently serve, in an online manner. While demand-aware networks may be significantly more efficient than demand-oblivious networks, frequent adjustments are still costly. Furthermore, a centralized controller of such networks may become a bottleneck.**

CBNet is based on concepts from self-adjusting data structures, and in particular, CBTrees [12]. CBNet gradually adapts the network topology toward the communication pattern in an online manner, i.e., without previous knowledge of the demand distribution. At the same time, *bidirectional semi-splaying* and counters are used to maintain state, minimize reconfiguration

Selected References

On the Complexity of Traffic Traces and Implications
Chen Avin, Manya Ghobadi, Chen Griner, and Stefan Schmid.
ACM SIGMETRICS, Boston, Massachusetts, USA, June 2020.

Survey of Reconfigurable Data Center Networks: Enablers, Algorithms, Complexity
Klaus-Tycho Foerster and Stefan Schmid.
SIGACT News, June 2019.

Toward Demand-Aware Networking: A Theory for Self-Adjusting Networks (Editorial)
Chen Avin and Stefan Schmid.
ACM SIGCOMM Computer Communication Review (**CCR**), October 2018.

Dynamically Optimal Self-Adjusting Single-Source Tree Networks
Chen Avin, Kaushik Mondal, and Stefan Schmid.
14th Latin American Theoretical Informatics Symposium (LATIN), University of Sao Paulo, Sao Paulo, Brazil, May 2020.

Demand-Aware Network Design with Minimal Congestion and Route Lengths
Chen Avin, Kaushik Mondal, and Stefan Schmid.
38th IEEE Conference on Computer Communications (**INFOCOM**), Paris, France, April 2019.

Distributed Self-Adjusting Tree Networks
Bruna Peres, Otavio Augusto de Oliveira Souza, Olga Goussevskaia, Chen Avin, and Stefan Schmid.
38th IEEE Conference on Computer Communications (**INFOCOM**), Paris, France, April 2019.

Efficient Non-Segregated Routing for Reconfigurable Demand-Aware Networks
Thomas Fenz, Klaus-Tycho Foerster, Stefan Schmid, and Anaïs Villedieu.
IFIP Networking, Warsaw, Poland, May 2019.

DaRTree: Deadline-Aware Multicast Transfers in Reconfigurable Wide-Area Networks
Long Luo, Klaus-Tycho Foerster, Stefan Schmid, and Hongfang Yu.
IEEE/ACM International Symposium on Quality of Service (**IWQoS**), Phoenix, Arizona, USA, June 2019.

Demand-Aware Network Designs of Bounded Degree
Chen Avin, Kaushik Mondal, and Stefan Schmid.
31st International Symposium on Distributed Computing (**DISC**), Vienna, Austria, October 2017.

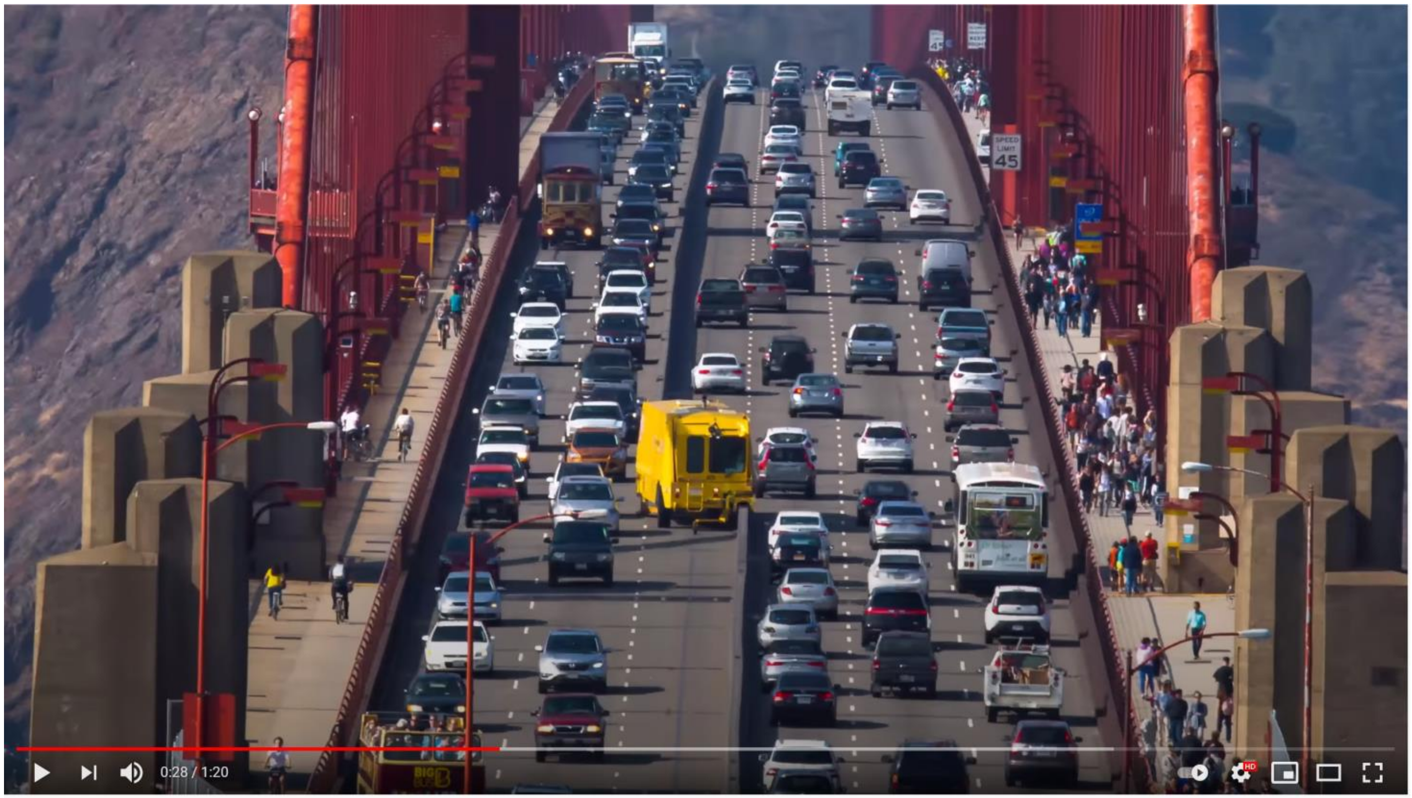
SplayNet: Towards Locally Self-Adjusting Networks
Stefan Schmid, Chen Avin, Christian Scheideler, Michael Borokhovich, Bernhard Haeupler, and Zvi Lotker.
IEEE/ACM Transactions on Networking (**TON**), Volume 24, Issue 3, 2016. Early version: IEEE **IPDPS** 2013.

Characterizing the Algorithmic Complexity of Reconfigurable Data Center Architectures
Klaus-Tycho Foerster, Monia Ghobadi, and Stefan Schmid.
ACM/IEEE Symposium on Architectures for Networking and Communications Systems (**ANCS**), Ithaca, New York, USA, July 2018.

Bonus Material

Hogwarts Stair

Bonus Material



Golden Gate Zipper

Bonus Material

IEEE SPECTRUM

Topics | Reports | Blogs | Multimedia | Magazine | Resources | Search

Tech Talk | Computing | Hardware

07 May 2021 | 16:55 GMT

Reconfigurable Optical Networks Will Move Supercomputer Data 100X Faster

Newly designed HPC network cards and software that reshapes topologies on-the-fly will be key to success

By **Michelle Hampson**

Photo illustration: Shutterstock

In HPC

Focus Topic:
Analysis of ProjecToR

Scheduling Opportunistic Links in Two-Tiered Reconfigurable Datacenters

Janardhan Kulkarni
Microsoft Research, Redmond, USA
jakul@microsoft.com

Stefan Schmid
University of Vienna, Austria
stefan_schmid@univie.ac.at

Paweł Schmidt
University of Wrocław, Poland
pawel.schmidt@cs.uni.wroc.pl

***Abstract*—Reconfigurable optical topologies are emerging as a promising technology to improve the efficiency of datacenter networks. This paper considers the problem of scheduling opportunistic links in such reconfigurable datacenters. We study the online setting and aim to minimize flow completion times. The problem is a two-tier generalization of classic switch scheduling problems. We present a stable-matching algorithm which is $2 \cdot (2/\varepsilon + 1)$-competitive against an optimal offline algorithm, in a resource augmentation model: the online algorithm runs**

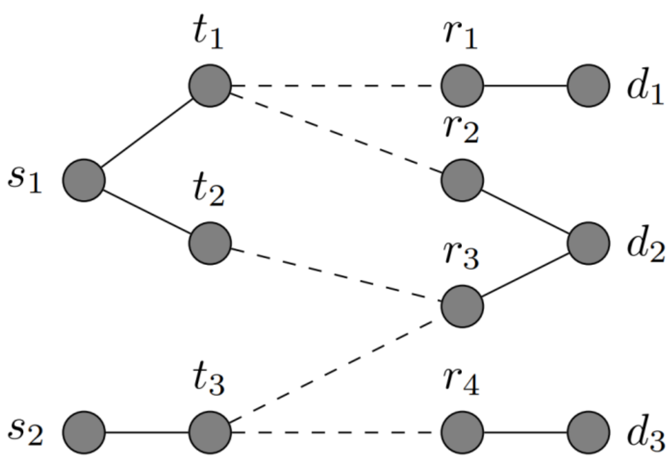
particular, we consider a two-stage switch scheduling model as it arises in existing datacenter architectures, e.g., based on free-space optics [11]. In a nutshell (a formal model will follow shortly), we consider an architecture where traffic demands (modelled as *packets*) arise between Top-of-Rack (ToR) switches, while opportunistic links are between lasers and photodetectors, and where many laser-photodetector combinations can serve traffic between a pair of ToRs. The goal is

ACM SPAA 2021

A 2-Tiered Architecture

Reconfigurable network of ProjecToR relies on a
2-tiered architecture:
⇢ Traffic demands (modelled as packets) arise between ToR switches
⇢ **Opportunitstic links** are between lasers and photodetectors

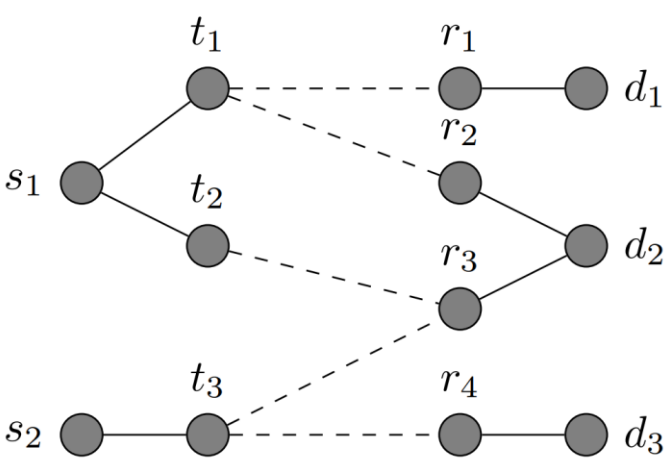
Many **laser-photodetector combinations** can serve traffic
between a pair of ToRs



How to optimally transmit packets over reconfigurable links (a matching)?

The Model

Packets between ToRs arrive in an
online fashion (adversarial).
Online **matching schedule** minimizing **latency**?



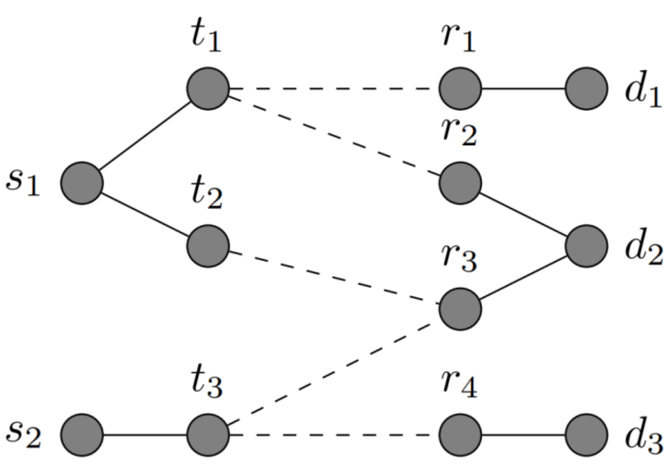
Round	Packet	Path	Latency
1st	S1 → D2	(T2,R3)	1
1st	S2 → D2	(T3,R3)	2
2nd	S2 → D2	(T3,R3)	2
2nd	S1 → D2	(T1,R2)	1
2nd	S1 → D1	(T1,R1)	2

Matchings:
1st round: (T2,R3)
2nd round: (T3,R3), (T1,R2)
3rd round: (T3,R3), (T1,R1)

ALG = 8
OPT = 6
use (T1,R2) and (T3,R3) in 1st round

The Model

Packets between ToRs arrive in an
online fashion (adversarial).
Online **matching schedule** minimizing **latency**?



Round	Packet	Path	Latency
1st	S1 → D2	(T2,R3)	1
1st	S2 → D2	(T3,R3)	2
2nd	S2 → D2	(T3,R3)	2
2nd	S1 → D2	(T1,R2)	1
2nd	S1 → D1	(T1,R1)	2

Matchings:
1st round: (T2,R3)
2nd round: (T3,R3), (T1,R2)
3rd round: (T3,R3), (T1,R1)

ALG = 8
OPT = 6
use (T1,R2) and (T3,R3) in 1st round

Related to online switch scheduling but 2 tiers!

Algorithm

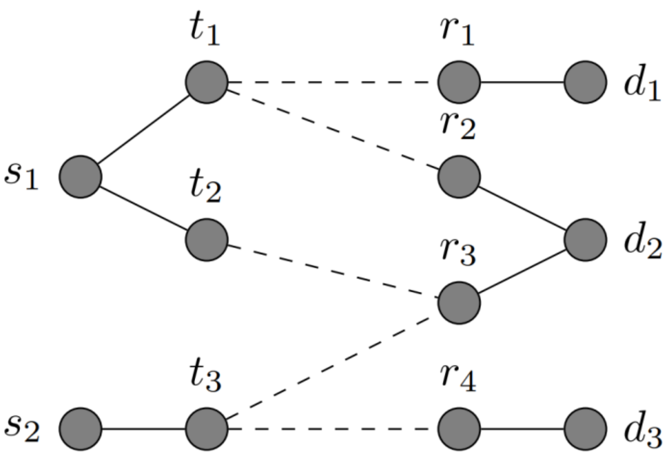
Scheduler ("transmit **stable matchings**"):

- ⇢ Based on a generalization of the stable-matching algorithm for two-tier networks
- ⇢ Each transmitter maintains a queue of packets that are not scheduled yet
- ⇢ In each time step, find a stable matching between transmitters and receivers

Dispatcher (**greedy**):

- ⇢ Incoming packet assigned to (transmitter, receiver) pair based on estimated worst case **latency increase**, taking into account set of queued packets in the system

Impact



Name	Path	Weight
A	(T1,R1)	2
B	(T1,R2)	8
C	(T1,R2)	9
D	(T3,R3)	4
E	(T3,R3)	7

What is the worst-case impact of S1 → D2 packet of weight 5?

Use (T1,R2): 5 + 5 + 2
Transmitted after BC, before A
Use (T2,R3): 5 + 4
Transmitted after E, before D

Result and Analysis

Alg is $O(\varepsilon^{-2})$-competitive in a resource augmentation model with speedup $(2+\varepsilon)$.

- Our algorithm is competitive in the **speed augmentation** model: online algorithm can transmit the packets at twice the rate of the optimal offline algorithm
- Dinitz and Moseley: otherwise **no competitive** online algorithm
- Analysis via **dual fitting** fitting inspired by scheduling for **unrelated machines**: take LP and dual-LP, assuming entire input (ok as only for analysis)
- The **crux** of the dual-fitting analysis: how to relate the cost of our algorithm to a feasible dual solution

High-Level Overview

- Find **LP relaxation** (primal) of the resource-augmented problem (OPT has limited transmission speed)
- Write its **dual** linear program
- Construct a **solution D** to dual program
- **Charge** ALG's cost to D
- Use **weak duality** to relate cost of D and OPT

Primal Program

Variables:

X_{peT} for packet p, compatible edge e, time T
(fraction of p sent through e at time T)

Objective:

$\Sigma_p \Sigma_e \Sigma_T$ weight(p) * (T - release(p)) * X_{peT}

Constraints:

$\Sigma_p \Sigma_e \Sigma_T \; X_{peT} \geq 1$

$\Sigma_e \Sigma_p \quad X_{peT} \leq 1/(2+\varepsilon)$ matching for transmitters

$\Sigma_e \Sigma_p \quad X_{peT} \leq 1/(2+\varepsilon)$ matching for receivers

Dual Program

Variables:

A_p for packet p

B_{tT}, B_{rT} for time T, transmitter t or receiver r

Objective:

$\Sigma_p A_p - 1/(2+\varepsilon)*(\Sigma_t \Sigma_T B_{tT} + \Sigma_t \Sigma_T B_{tT})$

Constraints:

$A_p - B_{tT} - B_{rT} \leq \text{weight}(p) * (T - \text{release}(p))$

For packet p, compatible edge $e = (t,r)$, time t

Dual Assignment

Variables:

A_p = worst-case impact of p

B_{tT} = weight of packets assigned to t, pending at time T

B_{rT} = weight of packets assigned to t, pending at time T

ALG-to-DUAL ratio:

$\Sigma_p A_p = \text{ALG}$

$\Sigma_t \Sigma_T B_{tT} = \Sigma_t \Sigma_T B_{tT} = \text{ALG}$

$$\text{DUAL} = \Sigma_p A_p - 1/(2+\varepsilon) * (\Sigma_t \Sigma_T B_{tT} + \Sigma_t \Sigma_T B_{tT})$$
$$= \text{ALG} - \text{ALG} * 2/(2+\varepsilon)$$
$$= \text{ALG} * \varepsilon/(2+\varepsilon)$$

Dual Assignment

Variables:

A_p = worst-case impact of p

B_{tT} = weight of packets assigned to t, pending at time T

B_{rT} = weight of packets assigned to t, pending at time T

ALG-to-DUAL ratio:

$$\Sigma_p A_p = ALG$$

$$\Sigma_t \Sigma_T B_{tT} = \Sigma_t \Sigma_T B_{tT} = ALG$$

$$\begin{aligned} DUAL &= \Sigma_p A_p - 1/(2+\varepsilon)*(\Sigma_t \Sigma_T B_{tT} + \Sigma_t \Sigma_T B_{tT}) \\ &= ALG - ALG * 2/(2+\varepsilon) \\ &= ALG * \varepsilon/(2+\varepsilon) \end{aligned}$$

It remains to bound **DUAL-to-OPT** ratio…

Analysis: DUAL-to-OPT

At time T = release(p) constraint

$A_p - B_{tT} - B_{rT} \leq$ weight(p) * (T − release(p))

holds with equality.

In one time step

LHS increases by 2 * weight(p)

(each B decreases by weight(p)

RHS increases by weight(p)

Halving each variable yields a **feasible solution** to **dual** program.

The Competitive Ratio

We have ALG = $(2+\varepsilon)/\varepsilon$ * DUAL

It remains to bound DUAL-to-OPT ratio

Halving each variable yields a feasible solution to dual program.

DUAL/2 $\leq$ OPT by weak duality

We obtain:

ALG $\leq 2 * (2+\varepsilon)/\varepsilon *$ **OPT**

Supported Extensions

- Different edge lengths in the network
- Packets sizes
- **Hybrid** fixed and reconfigurable networks

Focus Topic:
Relationship to Spanners

Demand-Aware Network Designs of Bounded Degree*

Chen Avin[1], Kaushik Mondal[1], and Stefan Schmid[2]

1 Communication Systems Engineering Department
Ben Gurion University of the Negev, Israel
avin@cse.bgu.ac.il, mondal@post.bgu.ac.il

2 Department of Computer Science
Aalborg University, Denmark
schmiste@cs.aau.dk

Abstract

Traditionally, networks such as datacenter interconnects are designed to optimize worst-case performance under *arbitrary* traffic patterns. Such network designs can however be far from optimal when considering the *actual* workloads and traffic patterns which they serve. This insight led to the development of demand-aware datacenter interconnects which can be reconfigured depending on the workload.

Motivated by these trends, this paper initiates the algorithmic study of demand-aware networks (DANs), and in particular the design of bounded-degree networks. The inputs to the network

DISC 2017

Low-Distortion Spanners

- Classic problem: find sparse, distance-preserving (low-distortion) **spanner** (the “DAN”) of a graph (the demand)
- ***But:***
  - Spanners aim at low distortion among all pairs; in our case, we are only interested in the **local distortion**, 1-hop communication neighbors
  - We allow **auxiliary edges** (not a subgraph): similar to geometric spanners
  - We require **constant degree**

Still Exploitable!

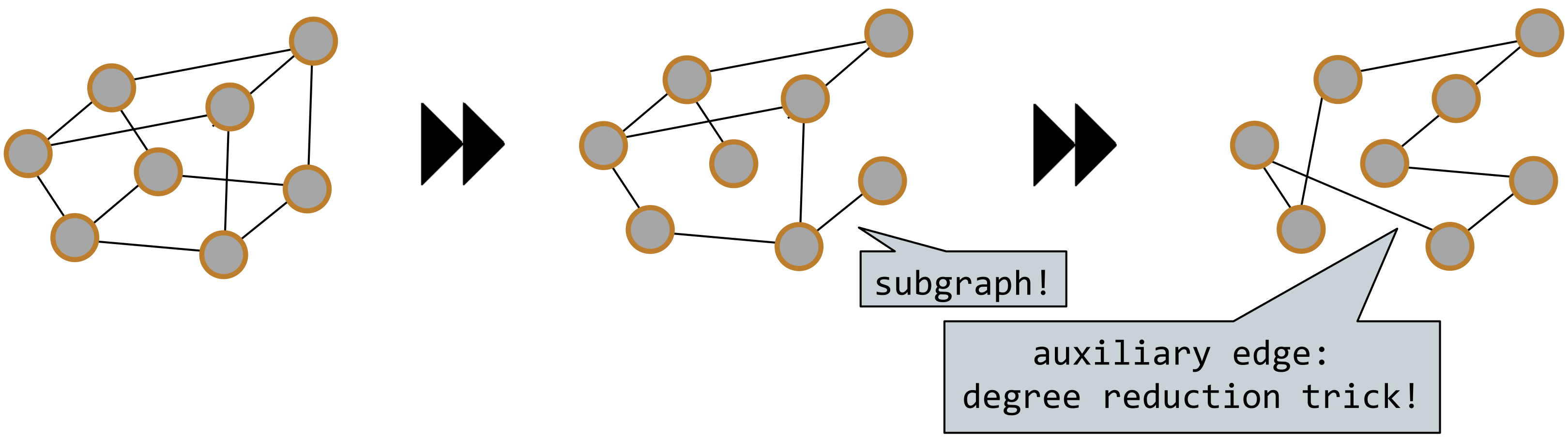
⇢ Yet: Can sometimes leverage connection to spanners

Theorem: If request distribution $\mathcal{D}$ is **regular and uniform**, and if we can find a constant distortion, linear sized (i.e., **constant sparse**) spanner for this request graph: then we can design a constant degree DAN providing an **optimal ERL** (i.e.,$O(H(X|Y)+H(Y|X))$.

r-regular *and* ***uniform*** **demand**:

Sparse, ***irregular*** ***(constant)*** **spanner**:

Constant *degree* ***optimal*** **DAN** (ERL at most ***log r***):



Still Exploitable!

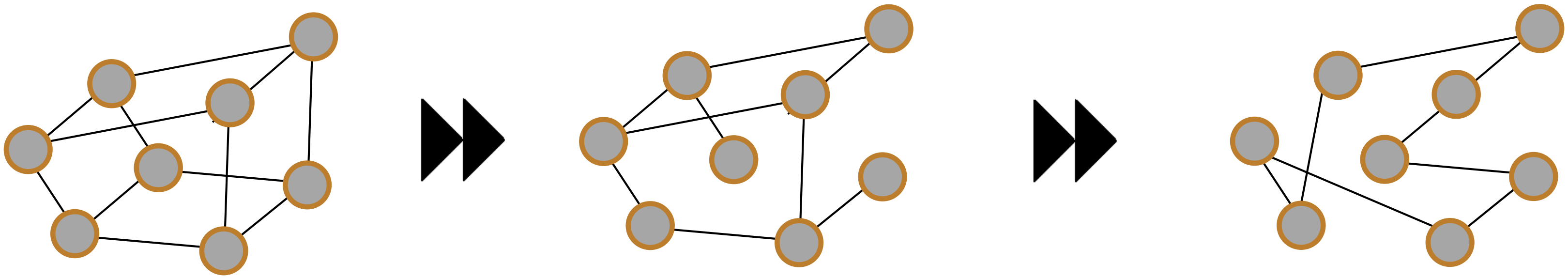
⇢ Yet: Can sometimes leverage connection to spanners

Theorem: If request distribution $\mathcal{D}$ is **regular and uniform**, and if we can find a constant distortion, linear sized (i.e., **constant sparse**) spanner for this request graph: then we can design a constant degree DAN providing an **optimal ERL** (i.e.,O(H(X|Y)+H(Y|X)).

r-regular *and* ***uniform*** **demand**:

Sparse, ***irregular*** ***(constant)*** **spanner**:

Constant *degree* ***optimal*** **DAN** (ERL at most ***log r***):



⇢ Optimality: r-regular graphs have entropy log r.

Corollaries

Optimal DAN designs for

⇢ Hypercubes (with n log n edges)

⇢ Chordal graphs

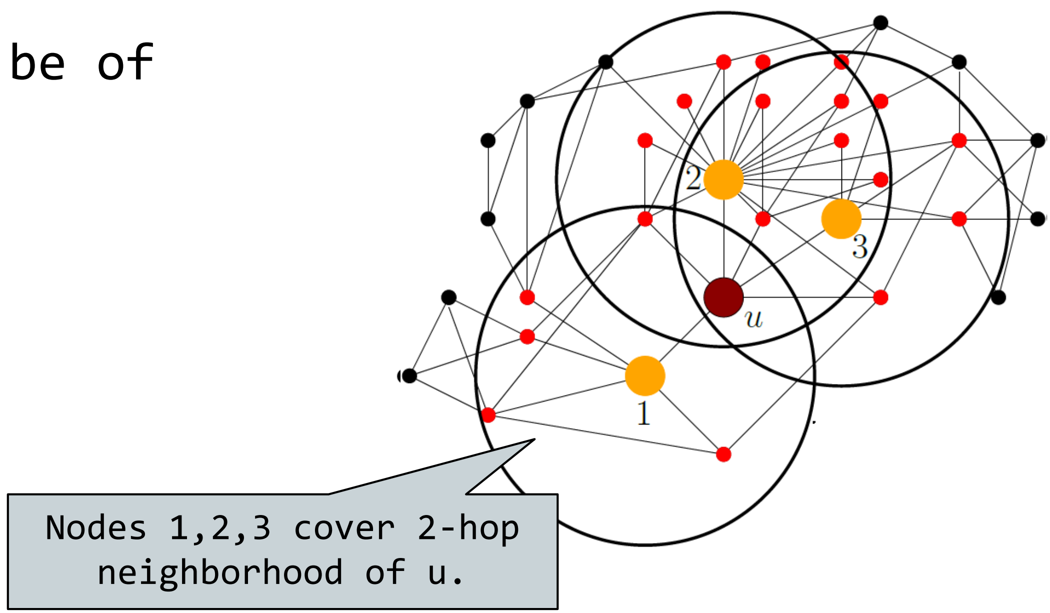
⇢ Trivial: graphs with polynomial degree (dense graphs)

⇢ Graphs of locally bounded doubling dimension

Example

- Definition: Demand graph has a Locally-bounded Doubling Dimension (LDD) iff all 2-hop neighbors are covered by 1-hop neighbors of just λ nodes
  - Note: care only about **2-neighborhood**

- Challenge: can be of **high degree**!



Example

Lemma: There exists a sparse 9-(subgraph)spanner for LDD. This implies optimal DAN: still focus on regular and uniform!

Def. (**ε-net**): A subset V' of V is a ε-net for a graph G = (V,E) if

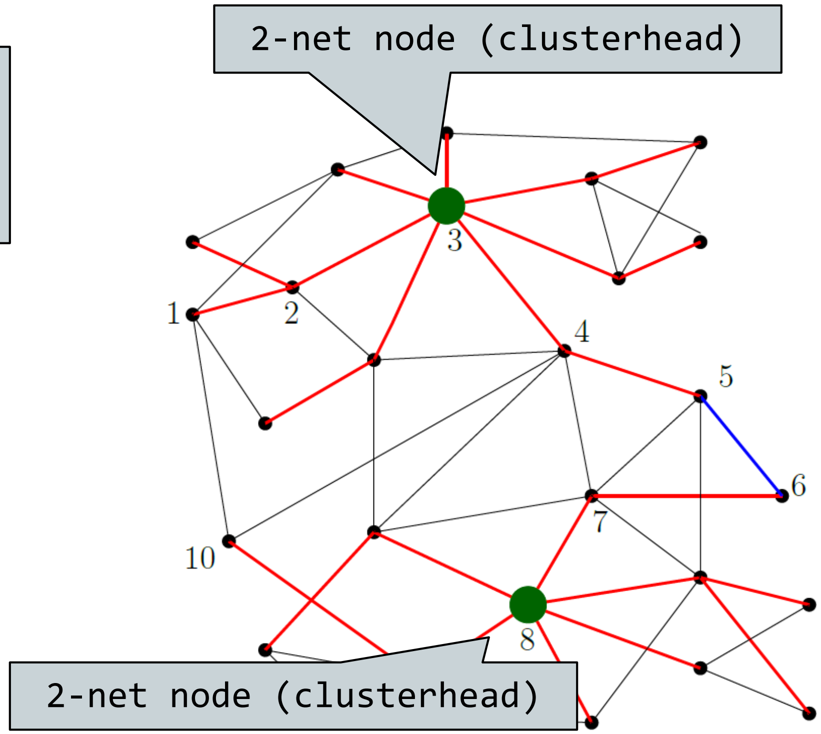
- V' sufficiently "**independent**": for every $u,v \in V'$, $d_G(u, v) > \varepsilon$
- "**dominating**" V: for each $w \in V$, $\exists$ at least one $u \in V'$ such that, $d_G(u,w) \leq \varepsilon$

Example

Simple algorithm:

1. Find a 2-net

Easy: Select nodes into 2-net **one-by-one** in decreasing (remaining) degrees, **remove 2-neighborhood**. Iterate.



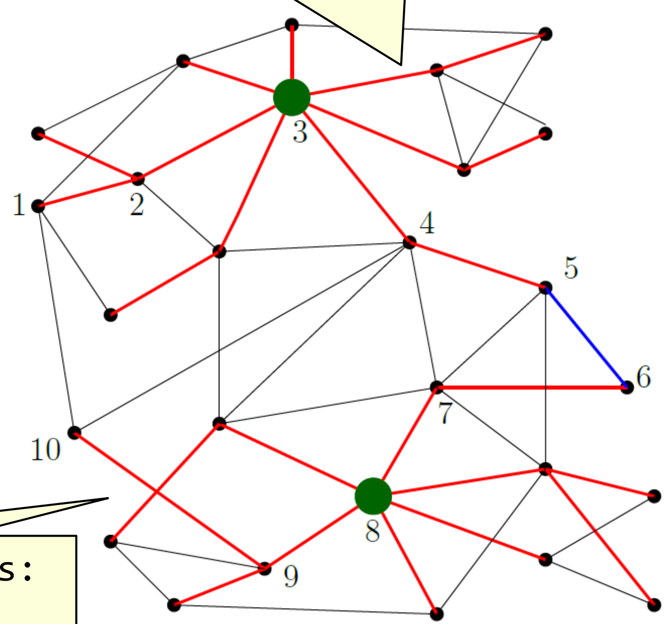
Example

Simple algorithm:

1. Find a 2-net

2. Assign nodes to one of the closest 2-net nodes: tree

Assign: at most 2 hops.



Union of these shortest paths:
a forest. Add to spanner.

Example

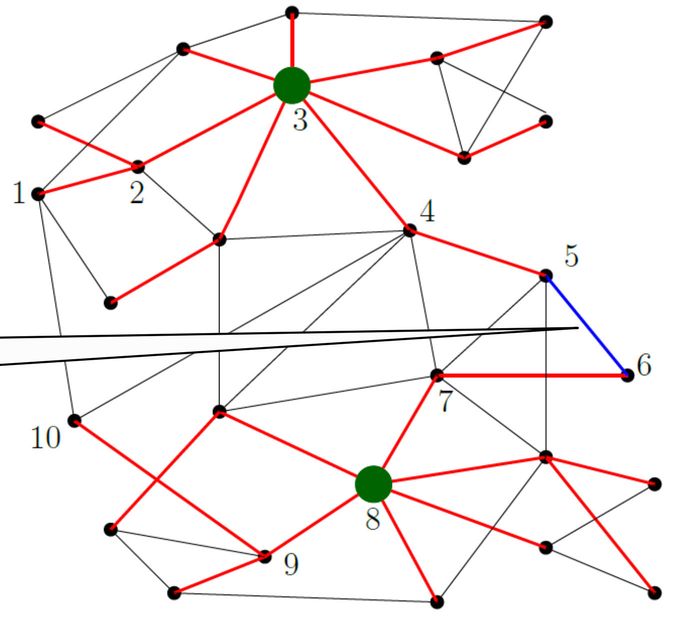
Simple algorithm:

1. Find a 2-net

2. Assign nodes to one of the closest 2-net nodes: tree

3. Join two clusters if there are edges in between

Connect forests (single „**connecting edge**“): add to **spanner**.



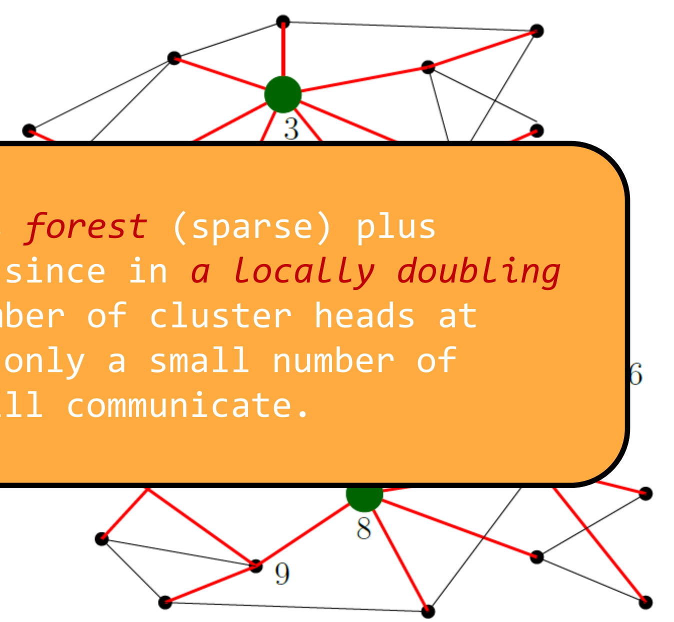
Example

Simple algorithm:

1. Find a 2-net

Distortion 9: *Short detour* via clusterheads: u,ch(u),x,y,ch(v),v

3. Join two clusters if there are edges in between



Sparse: Spanner only includes *forest* (sparse) plus “connecting edges”: but since in *a locally doubling dimension graph* the number of cluster heads at distance 5 is bounded, only a small number of neighboring clusters will communicate.